Constrained Generation and Adaptive Selection of C-Tests

vom Fachbereich Informatik
der Technischen Universität Darmstadt
genehmigte

Dissertation

zur Erlangung des akademischen Grades
Doctor rerum naturalium
(Dr.rer.nat.)

vorgelegt von
Ji-Ung Lee

Erstgutachterin: Prof. Dr. Iryna Gurevych
Zweitgutachter: Prof. Dr. Torsten Zesch

Darmstadt 2024

Lee, Ji-Ung: Constrained Generation and Adaptive Selection of C-Tests

Tag der Einreichung: 07.05.2024
Tag der Disputation: 09.07.2024

Darmstadt, Technische Universität Darmstadt

Acknowledgements

I had left the acknowledgements as the last step for finalizing my Ph.D. thesis; thinking that this would be the most fun section to write. Now that I've started writing it, I'm not sure what to write as all the funny jokes have magically escaped my mind. However, I want to refer all future Ph.D. students to *PhD Comics*[1] and *sh*t my reviewers say*[2] for some consolidation and fun things about academia. In the end, this mostly ended up as a list of people I want to thank for helping me through my Ph.D. time.

First and foremost (I guess this is the default order?), I want to thank both my supervisors Iryna Gurevych and Christian M. Meyer. Somehow they always managed to strike the balance between push and pull—keeping me motivated throughout the Ph.D. and sane enough to finally finish it. Especially Christian was a great mentor, teaching me the ways of an academic in my days as a junior Ph.D. student and always finding the right words (especially when writing papers). Special thanks also go to Lisa Beinborn for sharing all her work on C-Tests that provided the foundation for this thesis.

Of course, I'm very thankful to all my colleagues from the lab for the constructive feedback that I received during various talks and discussions (and for all the fun at (non-)work related events and conferences). In particular, I would like to thank (random order): Andreas Rücklé, Nils Reimers, Ilia Kuznetsov, Edwin Simpson, Gözde Gül Şahin, Yang Gao, Benjamin Schiller, Nafise Sadat Moosavi, Thy Thy Tran, Hendrik Schuff, Serwar Basch, Martin Tutek, Ajie Utama, Leonardo Ribeiro, Falko Helm, Nico Daheim, Haishuo Fang, Federico Tiblias, Cecilia Chen Liu, Hovhannes Tamoyan, Rachneet Sachdeva, Thomas Arnold, Ivan Habernal, Luke Bates, Irina Bigoulaeva, Jan Buchmann, Dennis Zyska, Nils Dycke, Max Eichler, Dominic Petrak, Qian Ruan, Sukannya Purkayastha, Gisela Vallejo, Mohsen Mesgar, Tristan Miller, Christian Stab, Imbesat Hassan Rizvi, Indraneil Paul, Sebastian Ochs, Lena Held, Sheng Lu, Aishik Mandal, Anmol Goel, Timour Igamberdiev, Yongxin Huang, Justus-Jonas Erker, Richard Eckart de Castilho, Beto Boullosa, Jonathan Tonglet, Chris Stahlhut, Steffen Eger, Johannes Daxenberger, Kexin Wang, and everyone else I forgot to mention (Sorry!).

Besides Christian and Lisa, I also thank all my other colleagues with whom I had the pleasure to share the same two-person office but with four people: Claudia Schulz, Jonas Pfeiffer, Christopher Klamm, Gregor Geigle, Tobias Mayer, Aniket Pramanick, Haritz Puerto, Tim Baumgärtner, and Furkan Şahinuç (though only for one day). Of course, this was all done to show the efficiency of min-*space think-tanking*, and totally in a scientific spirit in our strive for excellence, to advance existing borders and to explore new horizons (not because we lack space). Many thanks also go to our office admins Sue Messenger, Elisabeth Niemann, Denise Switalla, and Silvia Previero as well as our sysadmins Sebastian Alles, Fabian Kaiser, and Ute Winchenbach. Thank you for your patience and sorry for all the administrative work I may have caused unwillingly due to missing fields in travel forms or student helper contracts. A special thanks go to Tim Neubacher for all the fun gossip and to Max Glockner, a dear coffee buddy. Finally, I

[1] https://phdcomics.com/

[2] https://shitmyreviewerssay.tumblr.com/

want to thank Tilman Beck for the great jam sessions and all SIG Job members: Michael Bugert, Jan-Christoph Klie, and Yevgeniy Puzikov.

I would like to thank my loving parents Jong-Yol and Kum-Sun, and my dear sister In-Yeong for your love and support. Although we did not have much growing up, you always ensured that we could live with joy and dignity for which I am really grateful. My achievements were only possible due to your sacrifices and you always being by my side. I also thank my friends from school and town, particularly: Pia Fleckenstein, Christian Unkelbach, Sebastian Trippel, Matthias Kerekes, Nicolay Mohammed-Hadi, Julian Rosenberg, Thomas and Markus Olschewski, Fabian Korzer, Nicolas Mauder, and especially Daniel Merget who will be missed dearly. A special shout-out goes to all my band colleagues, all members of the organizing team of Darmstadt kocht, and the long-term Pathfinder group (we will finish our adventure eventually, despite our unfathomably low dice rolls). Many thanks also go to my friends from university, especially to Daniel Lehmann, Magnus Brand, Tim Kranz, Mark Prediger, and Marvin Dickhaus for all the geeky and nerdy discussions. I especially want to thank my friends Alex Burkl and Sebastian Ehmes for being amazing flat mates. I hope we will continue to meet.

Lastly and most importantly, I want to express my deepest gratitude to Diana Brand, the love of my life. Thank you dearly for being there throughout the most stressful phase of my Ph.D. and for your continuous support and cheering. I am really thankful for you to have found me and am looking forward to our journey onwards.

Sincerely,
Ji-Ung Lee

Erklärungen laut Promotionsordnung

§8 Abs. 1 lit. c PromO

Ich versichere hiermit, dass die elektronische Version meiner Dissertation mit der schriftlichen Version übereinstimmt.

§8 Abs. 1 lit. d PromO

Ich versichere hiermit, dass zu einem vorherigen Zeitpunkt noch keine Promotion versucht wurde. In diesem Fall sind nähere Angaben über Zeitpunkt, Hochschule, Dissertationsthema und Ergebnis dieses Versuchs mitzuteilen.

§9 Abs. 1 PromO

Ich versichere hiermit, dass die vorliegende Dissertation selbstständig und nur unter Verwendung der angegebenen Quellen verfasst wurde.

§9 Abs. 2 PromO

Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Darmstadt, 07.05.2024

Ji-Ung Lee

Wissenschaftlicher Werdegang des Verfassers[3]

Okt 2010 – Okt 2013	Bachelor of Science (B.Sc.) in Informatik, Technische Universität Darmstadt.
Okt 2013 – Apr 2017	Master of Science (M.Sc.) in Informatik, Technische Universität Darmstadt.
Sep 2017 – heute	Doktorand, Ubiquitous Knowledge Processing (UKP-Lab), Technische Universität Darmstadt.

[3]Gemäß §8 Abs. 1 lit. a der Promotionsordnung der TU Darmstadt

Zusammenfassung

Globalisierung und Migration erhöhen zunehmend den Bedarf an Mehrsprachigkeit, welche bereits jetzt schon als eine der Schlüsselkompetenzen für Erfolg gilt. Institutionen wie die Europäische Kommission und das Hochkommissariat der Vereinten Nationen für Flüchtlinge projizieren bereits jetzt schon einen Zuwachs an Geflüchteten durch eine Verschärfung des Klimawandels. Obgleich dieser Entwicklung gibt es einen beständigen ein Mangel an Dolmetscher:innen. Gleichzeitig stellen verfügbare technische Lösungen, wie automatisierte Übersetzungsgeräte, keine adäquate Alternative dar, da diese—insbesondere für selten gesprochene Sprachen und spezifische Domänen wie Jura oder Medizin—stark hinter den Anforderungen zurückfallen. Ebenso schaffen Large Language Models (LLMs) keine Abhilfe, da diese anfällig gegenüber Stereotypisierungen und sogenannten Halluzinationen sind. Der konstante Mangel an qualifiziertem Lehrpersonal verschärft hierbei umso mehr den Mangel an Sprachlernmöglichkeiten. Diese könnten zwar durch Intelligent Tutoring Systems (ITS) bereichert werden, allerdings bedarf die Entwicklung solcher Systeme die Überwindung von hohen rechtlichen und bürokratischen Hürden aufgrund stark lizenzierter Daten und strengen Datenschutzauflagen. Dies macht Methoden notwendig, die dazu in der Lage sind, bereits aus geringem Feedback zu lernen.

Das Ziel dieser Thesis ist die Schaffung neuer Sprachlernmöglichkeiten durch die Entwicklung von Methoden, welche die Arbeit des Lehrpersonals erleichtern und neue Möglichkeiten des Selbststudiums anregen. Unser Anwendungsgebiet sind sogenannte C-Tests, Lückentexte, welche sich durch eindeutigere Lücken von den cloze tests abgrenzen. Hierfür werden im ersten Teil der Thesis neue Methoden zur automatischen Generierung solcher C-Tests entwickelt. Im Gegensatz zu vergangenen Arbeiten zeichnen sich unsere Methoden, welche auf Heuristiken und Constrained Optimization basieren, dadurch aus, dass sie C-Tests mit einer bestimmten Zielschwierigkeit erstellen können. Zudem garantiert unsere, auf Mixed-Integer-Programming basierende Methodik, dass spezifische Vorgaben während der C-Test Generierung eingehalten werden.

Im zweiten Teil dieser Arbeit entwickeln wir ein neues Samplingverfahren, um ein C-Test Selektionsmodell interaktiv zu trainieren. Hierfür verwenden wir Konzepte aus dem Bereich des Active Learning, welches darauf abzielt, nur Instanzen zu annotieren, welche optimal zum Modelltraining beitragen (Modell Objective). Auf den ersten Blick erscheint Active Learning ungeeignet für Sprachlernszenarien, da dies zur Selektion von Instanzen führt, die schwieriger zu annotieren sind—und dementsprechend auch unpassend für Lernende sind (d.h. zu schwierig oder zu einfach). Im Gegensatz hierzu steht die Selektion von C-Tests, welche mit großer Wahrscheinlichkeit für Lernende geeignet sind (User Objective); da diese nicht sonderlich hilfreich für das Modelltraining sind. Wir zeigen mit unserer Samplingstrategie, dass es möglich ist, Instanzen zu selektieren, welche beide Objectives gleichzeitig maximieren, und dass diese gleichzeitige Maximierung zu zur Selektion von C-Tests führt, die das Training und den Lernprozess optimieren; insgesamt sogar besser als die Optimierung der jeweiligen Einzelobjectives.

Im letzten Teil der Thesis widmen wir uns der Erschließung von interaktiven Annotationsszenarien als einen weiteren Anwendungsfall, welcher von der neuen Samplingstrategie profitieren könnte. Hierfür entwickeln wir zuerst eine Applikation, die am Anwendungsfall einer Prozesslernfabrik aufzeigt, wie der Arbeitsalltag von Angestellten durch die

interaktive Datenannotation erleichtert werden kann. Zuletzt zeigen wir, dass auch in Annotationsstudien Lernprozesse zu finden sind, und entwickeln Annotation Curricula, eine Methode zur Sortierung der annotierten Instanzen. Unsere Nutzungsstudie zeigt, dass Annotation Curricula die für die Annotation benötigte Zeit signifikant reduzieren.

Abstract

Increasing globalization and immigration is driving the importance of multi-lingual proficiency. Being able to communicate across different languages is already one of the key competencies that can define success—however, various institutions such as the European Council or the United Nations High Commissioner for Refugees predict that this trend will intensify even further with climate change and rising refugee numbers. Despite these concerning developments, a shortage of proficient human translators remains, while existing automated solutions fall far behind the requirements. For instance, current translation tools have been shown to perform substantially worse in low-resource languages or in specialized domains such as legal or medical—causing real-world harm through unreflected use. Large language models (LLMs) still exhibit biases and hallucinations—rendering them unreliable. At the same time, the continuous shortage of teachers leads to an increasing gap for language learning opportunities. While self-directed learning and intelligent tutoring systems (ITS) have the potential to alleviate some of the issues, research in this area suffers from limited available data—a result of proprietary software and data protection regulations. This calls for methods that are capable of learning efficiently from little user feedback.

The goal of this thesis is to provide new language learning opportunities by devising methods that alleviate the work for teachers and that empower learners to self-directed learning. For evaluation we use C-Tests, a type of gap filling exercise that is similar to cloze tests, but less ambiguous. In the first part of this thesis, we develop novel methods for generating C-Tests. In contrast to previous works, our methods—that are based on heuristics and constrained optimization—are capable of generating C-Tests with a specific target difficulty. Moreover, our method based on mixed-integer programming allows teachers to pose specific constraints which are guaranteed to be adhered, resulting in C-Tests that better suit their needs.

In the second part of this thesis, we devise a new sampling method to interactively train a C-Test selection model. We draw inspiration from active learning that aims to improve model training by only annotating instances that presumably help the model most (model objective). At first glance, active learning seems to be unfit for educational scenarios as it can lead to instances that are more difficult to annotate—or likewise, result in C-Tests that do not suit a learner's current proficiency. Conversely, only selecting instances that suit the learner's current proficiency—ideally with a high certainty (user objective)—will result in feedback that is uninformative for the model. We show that it is indeed possible to sample instances that optimize both and that this results in C-Tests which benefit model and learner better than sampling instances for each objective individually.

Finally, we explore interactive data annotation as a scenario that could benefit from our joint sampling strategy. We first develop an application that showcases the usefulness of interactive data annotation in a scenario where domain experts can interactively annotate data to ease their work. We then show how annotation studies in general comprise a learning process, and devise annotation curricula, a method to reorder annotated instances which significantly reduces annotation time.

Contents

Part I

Synopsis

Publications and My Contributions

This thesis is based on six scientific publications to which I contributed as the lead author. They were co-authored together with my advisors Iryna Gurevych and Christian M. Meyer, as well as following co-authors (in alphabetical order):

Betty van Aken, Niranjan Balasubramanian, Qingqing Cao, Manuel R. Ciosici, Leon Derczynski, Jesse Dodge, Jessica Zosa Forde, Nicholas Frick, Michael Hassid, Kenneth Heafield, Sara Hooker, Tianchu Ji, Jan-Christoph Klie, Pedro H. Martins, André F. T. Martins, Peter Milder, Joachim Metternich, Marvin Müller, Marc Pfetsch, Colin Raffel, Edwin Simpson, Noam Slonim, Erik Schwan, Roy Schwartz, Lorenz Stangier, Emma Strubell, Marcos Treviso, and Yuxi Wang.

I am grateful to all my co-authors and their significant contributions to these pleasant as well as successful collaborations. In the following, I describe my own contributions to each publication.

Core Publications

Chapter 3 corresponds to the following publications[4]:

> **Ji-Ung Lee**, Erik Schwan, and Christian M. Meyer. 2019. Manipulating the Difficulty of C-Tests. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*(ACL), pages 360–370, Florence, Italy.

Christian and I developed the idea for this work which was initially given as a Bachelor's thesis topic to Erik Schwan, whom Christian and I co-supervised. We developed the proposed algorithms together over the course of the thesis. I collected and prepared the datasets and provided an updated feature extraction system that was initially developed by Beinborn (2016). Erik implemented the proposed algorithms and conducted reproduction and ablation studies. He further recruited participants for the user study. I devised the study design with advice from Christian and implemented the study interface. I conducted experiments regarding the C-Test variability and provided an in-depth analysis of the user study. I wrote the initial draft of the article and performed the subsequent corrections. I discussed this work regularly with Christian and Erik, who helped me improve the draft.

> **Ji-Ung Lee**, Marc Pfetsch, Iryna Gurevych. 2024. Constrained C-Test Generation using Mixed-Integer Programming. *arXiv:2404.08821*.

[4] Works with a shared first authorship where each first author contributed equally are marked with *.

I conceived the research ideas after detailed discussions with Marc and Iryna. I performed all of the implementation work, planned and conducted all experiments, designed, implemented, and conducted the user study, and performed all of the analyses. I wrote the initial draft of the article and performed the subsequent corrections. I discussed this work regularly with Iryna and Marc, who helped me improve the draft.

Chapter 4 corresponds to the following publications:

> Marcos Treviso*, **Ji-Ung Lee***, Tianchu Ji*, Betty van Aken, Qingqing Cao, Manuel R. Ciosici, Michael Hassid, Kenneth Heafield, Sara Hooker, Colin Raffel, Pedro H. Martins, André F. T. Martins, Jessica Zosa Forde, Peter Milder, Edwin Simpson, Noam Slonim, Jesse Dodge, Emma Strubell, Niranjan Balasubramanian, Leon Derczynski, Iryna Gurevych, Roy Schwartz. 2023. Efficient Methods for Natural Language Processing: A Survey. *Transactions of the Association for Computational Linguistics*, 11: pages 826–860. MIT Press.

This survey paper is a joint effort of 22 researchers led by Marcos Treviso, Tianchu Ji, and me as equal contribution first authors. The idea for this work was developed at the Dagstuhl seminar 22232: *Efficient and Equitable Natural Language Processing in the Age of Deep Learning*. The structure and content of the work was drafted as a group activity during two sessions at the seminar, with me being responsible for writing §8.2. After the seminar, Marcos, Tianchu, and I took the lead in refining the initial draft into a paper ready for submission. After the first rejection, Marcos and I revised the paper together, including change requests brought forward by the reviewers and adding new works that were published in the meantime. Throughout this effort, everyone involved worked on the section concerning their focus area (in the initial draft) and continued to provide suggestions and changes across the whole paper.

> **Ji-Ung Lee**, Christian M. Meyer, and Iryna Gurevych. 2020. Empowering Active Learning to Jointly Optimize System and User Demands. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (ACL), pages 4233–4247, Online.

I developed this idea as a key part of my thesis to tackle the lack of training data in the educational domain. I formalized the sampling objectives, implemented and conducted the experiments as well as the analysis. I further wrote the initial draft of the article and performed subsequent corrections. I discussed this work regularly with Christian and Iryna, who helped me improve the draft.

Chapter 5 corresponds to the following publications:

> Lorenz Stangier*, **Ji-Ung Lee***, Yuxi Wang, Marvin Müller, Nicholas Frick, Joachim Metternich, and Iryna Gurevych. 2022. TexPrax: A Messaging Application for Ethical, Real-time Data Collection and Annotation. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing: System Demonstrations* (AACL), pages 9–16, Taipei, Taiwan.

Lorenz Stangier and I equally contributed as first authors to this publication. The core idea for this project described in §10.2 was developed by Marvin Müller and me. Lorenz was primarily responsible for implementing the application described in §10.3. Marvin, Yuxi, and Nicolas were responsible for recruiting and instructing the participants of our study, and ensuring the correct annotation of the data (cf. §10.4). I devised the setup for the user study and wrote the initial draft of the paper. I furthermore finalized and packaged the code and analyzed the data. The experiments were conducted by Lorenz and me. We discussed this work regularly with Joachim and Iryna, who helped us improve the draft.

> **Ji-Ung Lee***, Jan-Christoph Klie*, Iryna Gurevych. 2022. Annotation Curricula to Implicitly Train Non-Expert Annotators. *Computational Linguistics*, 48 (2): 343-–373.

Jan-Christoph Klie and I equally contributed as first authors to this publication. We developed the core ideas during several brainstorming sessions. We researched and wrote the introduction, background, definition, and conclusion jointly. We also made the subsequent corrections together. Jan-Christoph collected and selected the datasets for the simulation described in §11.4. He further implemented, executed, analyzed the experiments, and wrote the chapter. Based on the simulation results, I designed, implemented, and conducted the user study described in §11.5. I further analyzed the results and wrote the respective chapter. We discussed this work regularly with Iryna, who helped us improve the draft.

All research results of the aforementioned publications are documented in the present thesis, which is archived by the Universitäts- und Landesbibliothek Darmstadt.

Research Data and Software

Each publication—except for the survey in Chapter 8 which is available at the ACL Anthology (CC BY 4.0)—entails various resources comprising software, models, and research data that were created and used to obtain the respective results. In the following, we provide brief descriptions and links to each component that was published under open source or creative commons licenses. We further ensure the long-term preservation of the research data in accordance with the Deutsche Forschungsgemeinschaft's (German Research Foundation) "Principles for the Handling of Research Data"[5], and archive all research data including the used data splits, intermediate results, models and hyper-parameter tuning experiments in TU datalib.[6] This data repository has solely been created for archiving purposes and contains proprietary data which may not be shared outside TU Darmstadt or used for any other purpose than reproducing the results of the respective publication. Any use of the C-Test data requires prior permission of the *Language Resource Centre* of TU Darmstadt.[7]

[5] https://www.dfg.de/download/pdf/foerderung/grundlagen_dfg_foerderung/forschungsdaten/leitlinien_forschungsdaten.pdf

[6] https://tudatalib.ulb.tu-darmstadt.de/handle/tudatalib/4207

[7] https://www.spz.tu-darmstadt.de/index.en.jsp

Chapter 6 Manipulating the Difficulty of C-Tests

The experiments conducted in this work comprise a reproduction study, an intrinsic evaluation of the new C-Test generation strategies, and a user study.

- All code to run our experiments is shared via GitHub (Apache 2.0):
  https://github.com/UKPLab/acl2019-ctest-difficulty-manipulation/
- The user study data and the models are shared via TU datalib (CC BY 4.0):
  https://tudatalib.ulb.tu-darmstadt.de/handle/tudatalib/2704
- The corresponding research data archive on TU datalib is:
  Chapter-6_ACL_2019_research_data.zip
- The publication is available at the ACL Anthology (CC BY 4.0):
  https://aclanthology.org/P19-1035/

Chapter 7 Constrained C-Test Generation via Mixed-Integer Programming

The experiments conducted in this work include a rework of the feature extraction pipeline, the implementation of multiple gap difficulty prediction models, and the reimplementation of the C-Test generation models from Chapter 6. In addition, we implemented multiple generation strategies using mixed-integer programming and conducted a user study.

- The code is structured into four segments corresponding to (1) feature extraction, (2) gap difficulty prediction, (3) C-Test generation, and (4) user study. All code is shared via GitHub (Apache 2.0):
  https://github.com/UKPLab/arxiv2024-constrained-ctest-generation
- The data comprises all results of our user study and the data sampled from the GUM (Zeldes, 2017) corpus we used for our intrinsic evaluation. All data and models are shared via TU datalib (CC BY 4.0):
  https://tudatalib.ulb.tu-darmstadt.de/handle/tudatalib/4205
- The corresponding research data archive on TU datalib is:
  Chapter-7_Arxiv_2024_research_data.zip
  Due to their large size, all fine-tuned transformer models are archived in separate `.zip` files, starting with `Chapter-7_Arxiv_2024`.
- A preprint of the publication is available on arXiv (CC BY 4.0):
  https://arxiv.org/abs/2404.08821

Preprocessing the C-Test data used in this work (as well as in Chapters 6 and 9) requires running the feature extraction pipeline developed by Beinborn (2016), that was previously split across multiple components. With this work, we provide two executable `.jar` files for sentence scoring and feature extraction that comprise all individual steps, and a single bash script for running the full extraction pipeline. Note, that feature extraction involves setting up a `DKPro Home` environment which relies upon two proprietary resources for which prior permission needs to be obtained:

- The TreeTagger and Chunker are available at
  https://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/
- The Web1T resource found at https://catalog.ldc.upenn.edu/LDC2006T13

For reproduction experiments, the TU datalib archive also contains a fully working `DKPro Home` environment: `DKPro.zip`

Chapter 8 Efficient Methods for Natural Language Processing: A Survey

This is a survey paper that has no code or research data contributions.

- The publication is available at the ACL Anthology (CC BY 4.0):
  https://aclanthology.org/2023.tacl-1.48/

Chapter 9 Empowering Active Learning to Jointly Optimize System and User Demands

The experiments conducted in this work evaluate different active learning strategies which are implemented and shared via GitHub. We are not allowed to share the C-Test data.

- All code to run our experiments is shared via GitHub (Apache 2.0):
  https://github.com/UKPLab/acl2020-empowering-active-learning
- The models to simulate learners of different proficiency are also shared via GitHub in the `learner_models` folder.
- The corresponding research data archive on TU datalib is:
  `Chapter-9_ACL_2020_research_data.zip`
- The publication is available at the ACL Anthology (CC BY 4.0):
  https://aclanthology.org/2020.acl-main.390/

Chapter 10 TexPrax: A Messaging Application for Ethical, Real-time Data Collection and Annotation

The experiments conducted in this work comprise three data collection studies. To conduct these studies, a bot was implemented for providing label suggestions and accepting label corrections. In addition, a dashboard was implemented and used for participants to keep track of the collected data.

- The code is structured into two segments corresponding to (1) the recorder bot and (2) the dashboard. All code is shared via GitHub (Apache 2.0):
  https://github.com/UKPLab/aacl2022-TexPrax

- The data consists of sentence- and token-level annotations. We further provide fine-tuned classification models using GermanBERT (Chan et al., 2020). All data and models are shared via TU datalib (CC BY 4.0):
  `https://tudatalib.ulb.tu-darmstadt.de/handle/tudatalib/3534`
- To ease usage, we further share the data via huggingface datasets along with the necessary `.load()` function:
  `https://huggingface.co/datasets/UKPLab/TexPrax`
- The corresponding research data archive on TU datalib is:
  `Chapter-10_AACL_2022_research_data.zip`
- The publication is available at the ACL Anthology (CC BY 4.0):
  `https://aclanthology.org/2022.aacl-demo.2/`

Chapter 11 Annotation Curricula to Implicitly Train Non-Expert Annotators

The experiments conducted in this work comprise an intrinsic evaluation study using three datasets and a user study.

- All code to run the intrinsic experiments and the user study is shared via GitHub in the respective subfolders (Apache 2.0):
  `https://github.com/UKPLab/cl2022-annotation-curriculum`
- The user study data and the models are shared via TU datalib (CC BY 4.0):
  `https://tudatalib.ulb.tu-darmstadt.de/handle/tudatalib/2783`
  Note, that the original tweets need to be re-crawled as it is proprietary.
- The corresponding research data archive (including the original tweets) on TU datalib is:
  `Chapter-11_CL_2022_research_data.zip`
- The publication is available at the ACL Anthology (CC BY-NC-ND 4.0):
  `https://aclanthology.org/2022.cl-2.4/`

Other Publications

During my time as a Ph.D. student, I was fortunate to work with great researchers on various topics, some of which did not fit into this thesis. In the interest of completeness, I provide references to these papers:

Jan-Christoph Klie, **Ji-Ung Lee**, Kevin Stowe, Gözde Gül Şahin, Nafise Sadat Moosavi, Luke Bates, Dominic Petrak, Richard Eckart De Castilho, Iryna Gurevych. 2023. Lessons Learned from a Citizen Science Project for Natural Language Processing. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3594–3608, Dubrovnik, Croatia.

Haishuo Fang, **Ji-Ung Lee**, Nafise Sadat Moosavi, and Iryna Gurevych. 2023. Transformers with Learnable Activation Functions. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 2382–2398, Dubrovnik, Croatia.

Ji-Ung Lee, Haritz Puerto, Betty van Aken, Yuki Arase, Jessica Zosa Forde, Leon Derczynski, Andreas Rücklé, Iryna Gurevych, Roy Schwartz, Emma Strubell, Jesse Dodge. 2023. Surveying (Dis)Parities and Concerns of Compute Hungry NLP Research. *arXiv preprint arXiv:2306.16900.*

Ulf A Hamster, **Ji-Ung Lee**, Alexander Geyken, Iryna Gurevych. 2023. Rediscovering Hashed Random Projections for Efficient Quantization of Contextualized Sentence Embeddings. *arXiv preprint arXiv:2304.02481.*

Tilman Beck, **Ji-Ung Lee**, Christina Viehmann, Marcus Maurer, Oliver Quiring, and Iryna Gurevych. 2021. Investigating label suggestions for opinion mining in German Covid-19 social media. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (ACL), pages 1–13, Online.

Marvin Müller, **Ji-Ung Lee**, Nicholas Frick, Lorenz Stangier, Iryna Gurevych, and Joachim Metternich. 2021. Extracting problem related entities from production chats to enhance the data base for assistance functions on the shop floor. In *9th CIRP Global Web Conference – Sustainable, resilient, and agile manufacturing and service operations : Lessons from COVID-19* (Procedia CIRP), Volume 103, pages 231–236, Online.

Marianne Grace Araneta, Gülşen Eryiğit, Alexander König, **Ji-Ung Lee**, Ana Luís, Verena Lyding, Lionel Nicolas, Christos Rodosthenous, and Federico Sangati. 2020. Substituto–A Synchronous Educational Language Game for Simultaneous Teaching and Crowdsourcing. In *Proceedings of the 9th Workshop on NLP for Computer Assisted Language Learning* (NLP4CALL) pages 1–9, Online.

Steffen Eger, Gözde Gül Şahin, Andreas Rücklé, **Ji-Ung Lee**, Claudia Schulz, Mohsen Mesgar, Krishnkant Swarnkar, Edwin Simpson, and Iryna Gurevych. 2019. Text Processing Like Humans Do: Visually Attacking and Shielding NLP Systems. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics* (NAACL-HLT), pages 1634–1647, Minneapolis, USA.

Ji-Ung Lee, Steffen Eger, Johannes Daxenberger, and Iryna Gurevych. 2017. UKP TU-DA at GermEval 2017: Deep Learning for Aspect Based Sentiment Detection. In *Proceedings of the GermEval 2017 - Shared Task on Aspect-based Sentiment in Social Media Customer Feedback*, Berlin, Germany.

Chapter 1

Second Language Acquisition

Learning a (second) language is one of the key factors that can directly impact a person's success in life (Baldo et al., 2005). Speaking the language of the residing country is not only key to participating in daily and social life—it can even positively impact the work life; for instance, by increasing employability (Yow and Lim, 2019). Research shows that achieving L2[1] competency on a level to be acknowledged as a native speaker substantially increases the chances for a successful immigration (Piller, 2002) and that learning a second language can even positively impact the L1 competency; a concept also known as *multicompetence* (Cook, 1992). Major institutions, such as the European Council, have acknowledged the importance of learning languages as early as in 2002, calling for an action to "[teach] at least two foreign languages from a very early age" (Council of Europe, 2002, p. 20). However, the current progress in terms of multi-lingual proficiency falls behind their initial ambitions. Even 15 years after the call to action, a study of the European Union finds that "if English becomes the «lingua franca» of the DSM [Digital Single Market], more than 60% of the European population will be left behind and with high disparities between countries" (Rivera Pastor et al., 2017, p. 51). Although the actions to improve the current state in terms of multi-lingual proficiency are easy to identify—for instance, increasing the number of qualified personnel or strengthening the availability of open source educational resources (i.e., data and software)—they are non-trivial to implement. Even something as basic as the shortage of teaching personnel has been attributed to numerous reasons that differ between countries and education systems (Blanco et al., 2023). At the same time, the lack of access to language learning courses remains one of the major obstacles for refugees to immigrate successfully (Fundamental Rights Agency, 2023, p. 24). Finally, there is an increasing demand for language learning beyond Europe—due to globalization, digital communication, and immigration (García and Weiss, 2019).

1.1 Translation Tools to the Rescue?

Natural language processing (NLP) is a research field that can help to address existing issues on low language proficiency (Jurafsky and Martin, 2000). Especially subfields such as machine translation (MT) and automatic speech recognition (ASR) have broken down language barriers to some extent (Carvalho et al., 2023); and recent research around large language models (LLMs) is showing promising results (Kocmi et al., 2023).

[1] In language learning research, the native language of a person is referred to as the first (or L1) language and any other language learned afterwards as the second (or L2) language (Klein, 1986).

Nonetheless, actual tools often remain far from being practicable in daily life for various reasons. First of all, they still fall behind human (expert) translators and hence, are legally not liable (Gupta et al., 2023; Vieira et al., 2023). This makes them infeasible to deploy in critical domains such as health or law, where mistakes can cause serious harm. One such unfortunate example are asylum applications—where even human translators struggle (Scheffer, 1997; Berbel, 2020)—that were wrongfully rejected due to erroneous translations.[2] Moreover, a high disparity in terms of available parallel training data between low-resource languages such as Swazi or Igbo and high-resource languages such as English, German, or French amplifies existing issues on excluding minority communities (Stap and Araabi, 2023; Robinson et al., 2023). Despite recent incentives that aim to curate training data for low-resource languages, this is a labor-intensive process which will take time to bear fruit (Siminyu and Freshia, 2020; Salesky et al., 2023). Another issue is that trained models are expensive to deploy despite novel methods that improve model efficiency across various factors such as run time, memory consumption, and model size (Treviso et al., 2023). Consequently, high-quality translations require a stable network connection to a model available online or sufficient computational resources to run the model locally. Whereas all these issues may be resolved in the future as research advances, there exists one big advantage of achieving a high L2 proficiency over any MT system; namely, fostering social inclusion by seamlessly bridging the language gap.

The importance of speaking the local language has been shown across many factors. For instance, works found that a high L2 proficiency positively impacts the income of immigrants (McManus et al., 1983; Tainer, 1988; Chiswick, 1991), and can substantially reduce periods of unemployment and foster social integration (Delander et al., 2005). Others have found that speaking the same language does not only benefit the individual, but also the group as a whole by increasing productivity (Yow and Lim, 2019). Finally, works suggest that speaking the same language has been one of the key factors to advance societies, promote cultural growth, and develop shared norms (Smith, 2010; Smith et al., 2017; Gelfand et al., 2024). This shows that human second language learning is and will remain important.

1.2 NLP in Education

The use of computers in language learning has been considered for a long time and across a wide range of use cases—such as automated tutoring systems (Hart, 1981) and teacher assistance (Ahmad, 1985). This research area is often referred to as *computer-assisted language learning* (CALL; Marty 1981). Following its loose definition, CALL has been associated with various research topics over the years: interactive learning systems, vocabulary learning, developing learning theory, as well as intelligent tutoring systems are only a few examples of CALL (Levy, 1997). In this thesis, we will focus on a repeating topic within this broad body of works, namely, the development of machine learning *models* that are deployed in CALL systems. Here, we find various issues that remain open (Kohnke

[2] Date of access: March 12, 2024
https://www.theguardian.com/us-news/2023/sep/07/asylum-seekers-ai-translation-apps

et al., 2023)—despite recent advances in tasks such as automatic essay rating (Naismith et al., 2023; Yancey et al., 2023), automated error correction (Katinskaia and Yangarber, 2023; Loem et al., 2023), or explanation generation (Duenas et al., 2023). First and foremost, a correct or "reliable" behavior of models needs to be ensured (Kenning and Kenning, 1990). Especially in self-directed learning scenarios, erroneous generations can cause substantial harm as there are no teachers to correct a model's mistake (Tlili et al., 2023). Second, it is not easy to maintain a continuous adaptation of models to multiple users (learners and teachers with varying preferences and learning goals) or different scenarios, while ensuring a high grade of personalization (Hu et al., 2023). And finally, there is a need to move away from NLP-focused metrics—such as BERTScore (Zhang et al., 2020)—to evaluating models or their generated content in terms of educational usefulness and real-world users (Levy, 1997; Imperial and Madabushi, 2023).[3]

1.3 Challenges and Contributions

Overall, we find that globalization and immigration are increasingly driving the need for solutions that can resolve the gap in multi-lingual proficiency. Despite recent advances in MT and LLMs, a lack of adequate translation tools remains, not to speak of the lacking resources to even train them for over 7,000 languages (Salesky et al., 2023). A focal and more sustainable solution is to improve the multi-lingual proficiency among the citizenry by assisting second language learners and teachers. Developing automated approaches to do so would also combat the existing shortage of teachers and open source educational resources; leading to new self-directed learning opportunities. Existing approaches however suffer from two shortcomings. First, they are incapable of strictly adhering to constraints posed by teachers. Using neural generation methods that can generate outputs which are "nonsensical, or unfaithful to the provided source input" (Ji et al., 2023) can substantially harm the learning process. Second, existing methods rely upon a pre-defined behavior (e.g., via hand-crafted rules) or utilize trained (but static) models. Even if models are updated successively with learner feedback, they still may be too slow to adapt, leading to a low grade of personalization which can impede learning (Illeris, 2003). At the same time, the growing number of learners makes it increasingly difficult to consider all possible learning types.

The goal of this thesis is to break new ground in CALL by addressing these shortcomings. First, we devise methods that provide mathematical guarantees on the generated output, ensuring that the resulting exercise cannot harm the learning process. Second, we devise methods that explicitly consider a learner's proficiency and moreover, that are capable of keeping up with their progress. As our use case, we focus on C-Tests (Klein-Braley and Raatz, 1982), a specific kind of gap filling exercises that are based on a partial deletion of words (similar to cloze tests). Finally, we show that our methods are applicable not only within the scope of CALL, but go beyond it and demonstrate a huge potential in data annotation scenarios. Our contributions (**C**) are as follows:

[3] For instance, Baladón et al. (2023) show that the BERTScore metric can be tricked with a simple baseline that always replies "Hello" in a teacher response generation shared task (Tack et al., 2023).

C1 We are the first to explore automated methods for C-Test generation that deviate from the static generation strategy. Our novel generation strategies utilize heuristics, constrained optimization, and state-of-the-art difficulty prediction models to generate C-Tests of a specific target difficulty. We show in multiple user studies that our strategies significantly outperform the static generation strategy and current neural approaches while adhering to constraints posed by teachers.

C2 We propose a novel strategy for efficient model training in interactive scenarios and evaluate it on the task of learner-adaptive C-Test selection. While existing works focus on selecting C-Tests that best suit a learner (user objective), we argue that this results in instances that are not optimal to train the selection model (model objective). Consequently, the model only slowly adapts to changes in the learner's proficiency. In our experiments with simulated learners, we show that jointly optimizing both (user and model objectives) is more beneficial than optimizing each individually—resulting in C-Tests that benefit the learner and produce feedback for instances that optimally train the model.

C3 We identify data annotation as another field that can benefit from user-adaptive sampling strategies and educational approaches. To pave the way towards interactive data annotation, we showcase how labeling instances can be directly built into a domain-specific use case to ease the work of factory workers. Finally, we show how annotators can be implicitly trained—revealing parallels between annotation studies and learning applications—and propose annotation curricula as a means to significantly reduce the annotation time.

Throughout our work, we do not only rely upon automated metrics, but conduct multiple user studies that are carefully designed and controlled for possible confounding variables. We further share all collected data and code under open source licenses to create research opportunities for future work.

1.4 Thesis Outline

So far, we have motivated the importance of second language acquisition and discussed the inadequacy of up-to-date MT tools. We then identified challenges in existing methods with respect to their reliability, adaptivity, and evaluation; and finally, detailed our contributions. The remainder of this thesis is structured as follows.

Chapter 2 - C-Tests introduces the type of gap filling exercise that serves as our primary use case throughout this work. We compare them to other types of gap filling exercises; identifying advantages and shortcomings of C-Tests. Finally, we discuss two major theories that are used in educational research to assess the difficulty of exercises and define formulas for assessing the gap and exercise difficulty of C-Tests.

Chapter 3 - Constrained C-Test Generation identifies the key challenges we face when considering C-Test generation; namely, efficiency and mathematical guarantees. Finding that there is a substantial lack of methods that propose automated generation strategies for C-Tests, we introduce two novel approaches. Finally, we discuss how these

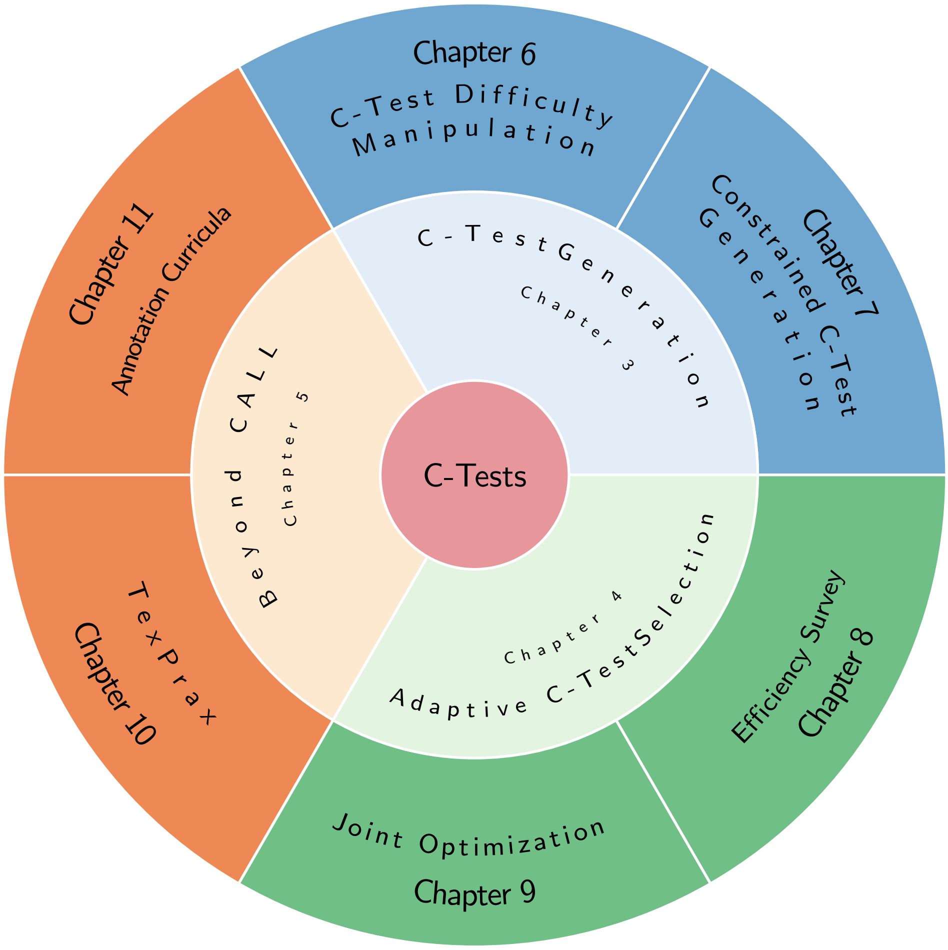


Figure 1.1: Overall structure of this thesis.

approaches alleviate the issues on hallucination, bias, alignment, and (to some extent) cost.

Chapter 4 - Adaptive C-Test Selection motivates the need for methods that are capable to adaptively select appropriate C-Tests as learners improve their proficiency. We discuss the notion of efficiency as a key component for adaptive selection methods and introduce active learning as a data efficient, interactive annotation paradigm. As active learning focuses on data annotation, we introduce the notion of a user objective and propose first suitable sampling strategies that consider both, model and user objective. Finally, we provide an overview of other exercise selection methods; finding that they only aim to optimize either one of the two.

Chapter 5 - Beyond CALL extends our work to data annotation as another scenario that requires interactive learning. We showcase how interactive data annotation can be of use especially in domains that require a high domain knowledge and where annotated data is scarce. We further show that learning takes place in even common data annotation scenarios and how devising a proper annotation curriculum can significantly reduce annotation time.

Over the course of this thesis, we developed and published multiple works that can be attributed to a respective chapter. Figure 1.1 provides an overview of the connection between different chapters and individual works in the context CALL. In the following, we list all publications in the same order as in Part I.

Chapter 6 - Manipulating the Difficulty of C-Tests introduces a first study on C-Test generation strategies that deviate from the commonly used, static one. Although the developed strategies are restricted to either manipulating the gap size or gap placement, our user study shows that both strategies can successfully generate C-Tests with a significantly different target difficulty.

Chapter 7 - Constrained C-Test Generation via Mixed-Integer Programming proposes an approach that considers the whole space of possible C-Tests and is capable of varying the gap size and placement. Using mixed-integer programming furthermore empowers the generation strategy to provide mathematical guarantees on the generated C-Test. More specifically, it upholds any (feasible) constraint such as the number of gaps or their size while achieving a globally optimal solution under a trained gap difficulty prediction model. Our evaluation study shows that our strategy significantly outperforms two of our baseline strategies (including GPT-4; OpenAI 2023) and performs similar to the third one.

Chapter 8 - Efficient Methods for Natural Language Processing: A Survey provides a comprehensive overview of methods that increase efficiency for corresponding steps in the life cycle of an NLP model. Most relevant for this thesis is §8.2 that discusses methods to improve data efficiency and which motivates active learning as a key concept to develop an efficient and adaptive C-Test selection method.

Chapter 9 - Empowering Active Learning to Jointly Optimize System and User Demands develops a sampling method that jointly considers a user's goal of obtaining the most appropriate C-Test and a system's goal of obtaining annotations for a C-Test that leads to a maximum improvement of the model. We find that jointly optimizing both goals—albeit seemingly counteractive—improves the overall performance over optimizing them individually.

Chapter 10 - TexPrax: A Messaging Application for Ethical, Real-time Data Collection and Annotation develops a system that allows us to transfer our research findings into a real-world use case. We discuss various necessities and their implementation to ensure a GDPR compliant data collection and develop a messenger application which satisfies all conditions. Using the resulting system and a label suggestion bot, we show how to effectively collect domain specific data in a real-world factory environment.

Chapter 11 - Annotation Curricula to Implicitly Train Non-Expert Annotators identifies data annotation as another scenario where the order in which instances are presented to a user significantly changes the outcome. In experiments with existing datasets and real-world users, we show that ordering instances according to their difficulty significantly reduces the overall annotation time. Besides an expert-based difficulty estimation, we further find that even a simple metric such as the BERTScore (Zhang et al., 2020) leads to a significant reduction of annotation time.

Finally, we conclude in Chapter 12 by summarizing the main research contributions of this thesis and discussing future research directions.

Chapter 2

C-Tests

Teachers need to consider a wide range of factors when they generate or select suitable language learning exercises. Such factors commonly involve the curriculum and depend on the already learned vocabulary, grammatical constructs, as well as the learner's proficiency (Kelly, 1977; Krashen, 1982). In addition, it is important to ensure that the exercises are neither too easy nor too difficult, as this may lead to boredom or frustration, respectively (cf. Figure 3.1). Exercises that keep the learner motivated are often ascribed to lie within the zone of proximal development (Vygotsky, 1978). Besides didactic factors such as the exercise difficulty and the learning curriculum, learner motivation can further be impacted by other factors such as a person's upbringing (Eccles, 2005, p. 106) and personal interest (Ainley, 2006). Simultaneously considering an arbitrarily large number of factors thus makes the manual generation of exercises very challenging. At the same time, copyright restrictions often limit the pool of tried and tested exercises teachers can select from (Prandner and Forstner, 2022). Automatically generating exercises is hence key to alleviate the work of teachers and to foster self-directed learning opportunities. Moreover, they can facilitate the growth of *open educational resources* (OER), increasing equity amongst learners (Hylén, 2020; Clinton-Lisell, 2021).

Gap filling exercises are a type of exercise that are especially well suited for automated generation for three reasons (Jongsma, 1971). First, their generation only requires partial deletion of an input text. Second, they allow teachers to control the difficulty, as they can decide how much (e.g., parts of a word, whole words, or even multiple words) and which parts to delete (e.g., by deleting frequently or infrequently used words). Finally, the input text can freely be chosen—for instance, from a book that is being discussed in class—easing their integration into a specific curriculum. Overall, gap filling exercises have been shown to be helpful for second language learning by fostering the reproduction, contextualization, and correct inflection of learned vocabulary (Oller Jr, 1973).

2.1 Cloze Tests

One of the most popular gap filling exercises are *cloze tests* (Taylor, 1953). Initially introduced to quantify text readability, they were increasingly utilized to measure second language proficiency (Oller Jr, 1973) and for vocabulary training (Skory and Eskenazi, 2010). Cloze tests always turn whole words into a gap; usually at a fixed deletion rate (e.g., a deletion rate of seven turns every seventh word into a gap). A shortcoming of the original cloze test is the high ambiguity caused by deleting whole words (Type $\mathcal{A}$,

Type	Example			Ambiguity	Automat-ability	Nature of Task
$\mathcal{A}$	The students have to	___	the test	high	high	production
$\mathcal{B}$	The students have to	___ a) take b) fold c) entertain d) fry	the test	low to moderate	low	recognition
$\mathcal{C}$	The students have to Their cook will All passengers should Both authors	___ ___ ___ ___	the test three salmons. their seats. credit for this.	low to moderate	high	production

Table 2.1: Different types of cloze tests (adapted from Wojatzki et al. 2016). Open cloze tests ($\mathcal{A}$) are easy to generate, but have a high ambiguity. Adding distractors ($\mathcal{B}$) substantially reduces the ambiguity, but is more difficult to automate as the distractors must to be incorrect. Moreover, this changes the nature of task to a simple recognition task as opposed to production. Finally, bundled gap filling ($\mathcal{C}$) uses multiple sentences with the same gap. This reduces the ambiguity and is easy to generate, but is limited to single sentences.

Table 2.1). These so-called *open cloze tests* only require the deletion of words but at the same time, pose a substantial challenge as words that lead to ambiguous solutions must be avoided (Felice and Buttery, 2019). To mitigate this, works soon proposed *multiple choice* formats (Type $\mathcal{B}$, Table 2.1) where the correct solution would be mixed within several incorrect *distractors* (Cranney, 1972). The difficulty of multiple-choice cloze tests can be controlled more explicitly than for open cloze tests; for instance, by selecting distractors according to their solution similarity. This and other approaches for distractor selection have been studied across the field (Zesch and Melamud, 2014; Hill and Simha, 2016; Yeung et al., 2019; Gao et al., 2020; Panda et al., 2022; Yoshimi et al., 2023). Finally, Wojatzki et al. (2016) propose *bundled gap filling* to mitigate the non-productive nature of multiple choice cloze tests (Type $\mathcal{C}$, Table 2.1). By displaying different sentences with the same solution at once, they have a substantially lower ambiguity compared to open cloze tests and require the learner to actively produce the correct solution. However, bundled gap filling has only been investigated for single gap sentences, as constructing similar exercises with multiple gaps per sentence or using whole paragraphs is substantially more difficult (Meyer et al., 2016).

2.2 C-Tests

As shown in Table 2.1, all variants of cloze tests suffer from one or more of the following issues:

- a high ambiguity of the gaps
- a low automatability

```
There are chance meetings with strangers that interest us from the first
moment, before a word is spoken.  Such w__(as) the impre_____(ssion) made
o_(n) Raskolnikov b_(y) the per___(son) sitting a lit___(tle) distance
fr__(om) him, w__(ho) looked li__(ke) a retired cl___(erk).  The yo___(ung)
man of___(ten) recalled th__(is) impression after_____(wards), and ev__(en)
ascribed i_(t) to presen______(timent).  He loo___(ked) repeatedly a_(t) the
cl___(erk), partly no doubt because the latter was staring persistently at
him, obviously anxious to enter into conversation.  At the other persons in
the room, including the tavern-keeper, the clerk looked as though he were used
to their company, and weary of it, showing a shade of condescending contempt
for them as persons of station and culture inferior to his own, with whom it
would be useless for him to converse.
```

Figure 2.1: A C-Test consisting of 20 gaps that has been generated by the commonly used static generation strategy where every second half of every second word is turned into a gap (Klein-Braley and Raatz, 1982). In this example, gaps are visualized by replacing the removed characters with "_". The solution is shown in parentheses "()" to display the original input text. The first and the last sentences do not contain any gaps to provide sufficient context.

- a non-productive nature of the task
- a limitation to single sentences

To obtain exercises that are easy to generate, have low ambiguity, are of productive nature, and can be generated from whole paragraphs, Klein-Braley and Raatz (1982) propose *C-Tests*. Similar to cloze tests, C-Tests are also based on deletion; but instead of the whole word, only its second half is turned into a gap, leaving the rest as a hint. This naturally reduces the ambiguity of the gaps and at the same time, still requires learners to actively inflect words themselves. Moreover, in contrast to bundled gap filling (Type $\mathcal{C}$, Table 2.1), they can span multiple sentences. For better comparability, C-Test creation follows a static generation strategy that turns every second half (rounded up) of every second word into a gap. The first and last sentences do not contain any gaps to provide sufficient context. Finally, the number of gaps is also pre-specified and commonly set to 20 or 25 (Grotjahn, 2006). Figure 2.1 shows an example C-Test consisting of 20 gaps that was created using the static generation strategy. As can be seen, only deleting the second half of a word substantially reduces the ambiguity. For instance, while the word "distance" provides a hint about the preceding gap "lit___(tle)", without the hint "lit" the word "far" would be an equally correct solution. Since their introduction, various works have shown that C-Tests are useful tools for L2 vocabulary acquisition (Chapelle, 1994; McKay, 2019) and moreover, that they also follow the reduced redundancy principle (Babaii and Ansary, 2001).[1] We thus choose C-Tests as our primary object of study.

[1]The concept of redundancy is rooted in information theory and quantifies the amount of information that can be removed in a message without removing any meaning (Shannon, 1948). Spolsky (1969) introduce redundancy as a concept to second language acquisition, finding that a learner's proficiency correlates with the capability to deal with reduced redundancy.

2.3 Automated Difficulty Assessment

Both—the generation and selection of exercises that suit a learner's language proficiency—require an automated assessment of their difficulty. The difficulty of gap filling exercises such as cloze tests and C-Tests can be assessed on two levels—on a gap level and an exercise level (for exercises that are comprised of multiple gaps). In the following, we will first discuss two theories from educational and psychological testing, that provide two fundamentally different views on modelling the difficulty. We will then provide an overview of existing methods that have been suggested for C-Test difficulty prediction and highlight our contributions in this area.

2.3.1 Modelling Difficulty

Quantifying the difficulty of specific items has been explored in both educational and psychological research. Traditionally, the term *item* refers to a single instance within a test or, in the case of gap filling exercises, to a single gap (Linden and Hambleton, 1997). Two main theories have been used to model the difficulty of single items and tests (or gaps and exercises, respectively): *classical test theory* and *item response theory*.[2]

Classical Test Theory. Classical test theory (CTT) is a well-established method to quantify the test difficulty (Novick, 1966; Lord and Novick, 1968). CTT provides a mathematical framework to describe the observed test score; i.e., the score a learner l achieved on a test t. Consider a gap filling exercise consisting of n gaps, where each gap can be answered either correctly (1) or incorrectly (0). The observed test score $\mathcal{X}_{l,t}$ is then defined as the fraction of correctly answered gaps:

$$\mathcal{X}_{l,t} = \frac{1}{n} \sum_{i=1}^{n} x_{l,i}, \tag{2.1}$$

where $x_{l,i}$ indicates if learner l has responded correctly at gap i. CTT then postulates that there exists a true score which represents the ability of a learner (i.e., the language proficiency) that is contained in the observed test score $\mathcal{X}_{l,t}$:

$$\mathcal{X}_{l,t} = \mathcal{T}_{l,t} + \mathcal{E}_{l,t}, \tag{2.2}$$

where $\mathcal{T}_{l,t}$ is the true score and $\mathcal{E}_{l,t}$ the error score. Although $\mathcal{T}_{l,t}$ and $\mathcal{E}_{l,t}$ are latent variables that cannot be measured, CTT postulates two additional assumptions that—together with Equation (2.2)—allow us to derive formulas for quantifying the exercise difficulty. First, $\mathcal{X}_{l,t}$ converges towards $\mathcal{T}_{l,t}$ if the test is repeated indefinitely. It directly follows that $\mathcal{E}_{l,t} \sim \mathcal{N}(0,1)$; i.e., the errors are normally distributed with zero mean. Second, $\mathcal{T}_{l,t}$ is independent of the test taken. This implies that taking an indefinite

[2]In the following, we use the terms *exercise* and *test* interchangeably as both refer to the same thing but in different contexts. Same goes for the terms *item* and *gap*.

number of measurements for a single test provides us with the true score of the test (across all learners). For k measurements we can thus say:

$$\mathcal{T}_{*,t} = \frac{1}{k} \sum_{j=1}^{k} \mathcal{T}_{j,t}, \tag{2.3}$$

where $\mathcal{T}_{j,t}$ is the true score of learner j for test t. Using the first assumption (i.e., $\mathcal{X}_{l,t}$ converges towards $\mathcal{T}_{l,t}$), we can put this in relation to the observed score as:

$$\mathcal{X}_{*,t} = \frac{1}{k} \sum_{j=1}^{k} \mathcal{X}_{j,t}, \tag{2.4}$$

which provides us with a formula for the test difficulty. For the difficulty of individual items $x_{*,i} \in \mathcal{X}_{*,t}$, we utilize the probability value (in short, the `P-value`) which CTT defines as:

$$x_{*,i} = \frac{1}{k} \sum_{j=1}^{k} x_{j,i}, \tag{2.5}$$

where i denotes the i-th item. Note, that CTT defines the probability value as the probability of the item being correctly solved.

Item Response Theory. The second well-established theory for quantifying test difficulty is the item response theory (IRT; Hambleton et al. 1991). In contrast to CTT that derives individual item difficulties from the test score (i.e., following a top-down approach), IRT derives the test difficulty from individual item scores (i.e., following a bottom-up approach). This is achieved by defining so-called item response functions (IRF) which are probabilistic models that express the probability of a learner correctly responding to an item. A key difference to CTT is that IRFs explicitly model a learner's ability θ (Hambleton and Jones, 1993; Linden and Hambleton, 1997). Over the years, multiple IRFs have been proposed to determine the item difficulty. The most basic form is the logistic model, also known as the Rasch model (Rasch, 1960):

$$P_{i,j}(x_{l,i} = \text{correct}) = \frac{\exp(\theta_l - \beta_i)}{1 + \exp(\theta_l - \beta_i)}, \tag{2.6}$$

where θ_l describes the ability of learner l, β_i is the difficulty of item i, and $x_{l,i}$ is the response of learner l for item i. To better model the characteristics of individual items, Birnbaum (1968) proposed to further extend Equation (2.6) with a discrimination parameter α_i:

$$P_{i,j}(x_{l,i} = \text{correct}) = \frac{\exp(\alpha_i(\theta_l - \beta_i))}{1 + \exp(\alpha_i(\theta_l - \beta_i))}. \tag{2.7}$$

In other words, if β_i expresses the difficulty of the i-th item, α_i expresses how well it can discriminate between different learners. The item difficulty is then defined as the ability required for a learner to solve the item with a probability of 0.5. It is important to note that the notion of item can be interpreted more fluently in IRT, compared to CTT. This has also motived researchers to consider whole tests as an item and utilize the underlying model for a learner-adaptive selection (Settles et al., 2020).

CTT vs IRT. Although IRT has gained increasing popularity over CTT in recent years, both theories have distinct advantages and disadvantages (Benedetto et al., 2023). IRT explicitly models individual learner abilities as well as item-specific characteristics. This allows researchers to make well-founded statistical adjustments in terms of the test score, which in turn can lead to a better selection of exercises for individual learners. In contrast, computing the test score is substantially easier in CTT and furthermore, the fraction of correct responses is much more intuitive to interpret than ability scores derived from a logistic function. Whereas recent works find that IRT and CTT have a high correlation and thus, can be used interchangeably (Setiawati et al., 2023), some research indicates that for small sample sizes, CTT may be a better choice (Mead and Meade, 2010). Similar observations have been made by Beinborn (2016) for C-Tests, who find a high correlation between IRT and CTT but less stable IRT estimates for small sample sizes.

C-Test Difficulty. In this thesis, we primarily rely upon the data provided by Beinborn (2016) to train our difficulty prediction models. We thus follow their suggestion and quantify the difficulty of a C-Test t as the mean error rate E across all learner responses:

$$E(t) = 1 - \frac{1}{n \cdot k} \sum_{j=1}^{k} \sum_{i=1}^{n} x_{j,i}, \tag{2.8}$$

where k is the number of learners, n the number of gaps in t, and $x_{j,i}$ the binary value (0/1) for the correctness.[3]

2.3.2 Predicting Difficulty

Past works have investigated various feature-based models, neural architectures, or combinations across different exercise types. Although our task at hand is a regression task, we also consider works that perform difficulty classification on an ordinal scale such as the CEFR-scale (Council of Europe, 2001); as both assume an underlying order that allows regression scores to be mapped into the CEFR-scale (Reichert et al., 2010). Overall, we find that the majority of works focus on the more popular cloze tests. Here, early works utilize lexical resources such as WordNet (Miller, 1995) together with regular expressions (Mitkov and Ha, 2003) but then follow overall trends in NLP research; moving towards n-gram models (Skory and Eskenazi, 2010; Hill and Simha, 2016; Felice and Buttery, 2019) and feature-based models (Liang et al., 2018). Finally, recent works also investigate the use of word embeddings (Hsu et al., 2018) and pre-trained Transformer models (Zhou and Tao, 2020; Benedetto et al., 2021).[4]

C-Test Difficulty Prediction. Besides early works that manually analyze C-Tests on a gap level (Kamimoto, 1993) or propose simple feature-based models to predict their difficulty (Sigott, 1995, 2006), Beinborn et al. (2014a) are the first to extensively study and advance the difficulty prediction of C-Tests. In multiple subsequent works, they

[3] Note, how $E(t)$ equals $1-$`P-value` from Equation (2.5) averaged across all items.

[4] We refer the interested reader to Benedetto et al. (2023) and AlKhuzaey et al. (2023) for recent surveys on difficulty estimation.

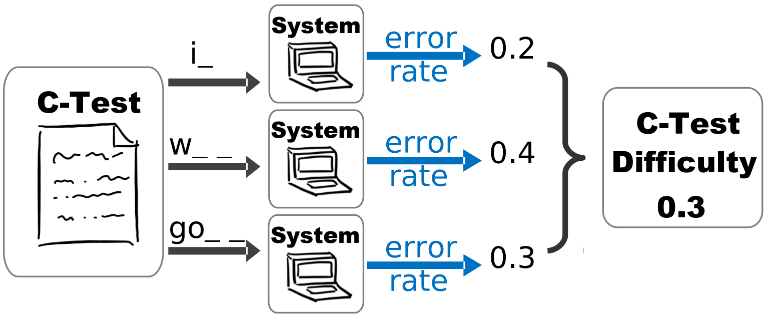


Figure 2.2: C-Test difficulty prediction. For each gap, a model predicts the respective difficulty which is then aggregated into the overall (mean) difficulty.

identify features across four categories: text difficulty, item dependency, word difficulty, and candidate ambiguity (Beinborn et al., 2014a, 2015; Beinborn, 2016). In addition, they establish Pearson's correlation (ρ) and root mean square error (RMSE) as evaluation metrics and use them to evaluate support vector machines (SVMs; Cortes and Vapnik 1995) and linear regression models. Besides these works, only Settles et al. (2020) study the difficulty prediction of C-Tests with IRT using linguistic features to train linear and weighted softmax models. The work is later extended by McCarthy et al. (2021) who also introduce BERT-based features and use the models for adaptive exercise selection. One substantial shortcoming of these works is the limited reproducibility due to the use of proprietary training and test data. Moreover, the code is not shared publicly, making it difficult to even consider these models for comparison.

2.4 Contributions

We make contributions towards C-Test difficulty prediction across three separate works. In Chapter 6, we tackle the long run time of the feature extraction pipeline proposed by Beinborn (2016). This is essential for C-Test generation which requires the repeated computation of features with varying gap sizes and placements. We thus perform ablation experiments to identify a subset of six features that are capable of estimating the relative change in difficulty with reasonable performance. This substantially increases the overall efficiency, allowing us to devise methods that iteratively increase or decrease the gap size. In Chapter 9, we improve the performance of the difficulty prediction model. More specifically, we identify two additional features derived from a pre-trained BERT model (Devlin et al., 2019): the prediction probability of the solution and the entropy computed using the top-50 tokens with the highest prediction probability (normalized using the softmax function). We further extend our evaluation to MLPs (Rosenblatt, 1958) and bi-directional LSTMs (Hochreiter and Schmidhuber, 1997). In Chapter 7, we re-evaluate all features for C-Tests with gap sizes and placements different from the static generation strategy (Klein-Braley and Raatz, 1982). Moreover, we propose three different fine-tuning mechanisms (masked-regression, `CLS`-token prediction, and feature-enriched `CLS`-token prediction) for transformer-based models and evaluate them using

BERT, RoBERTa (Liu et al., 2019), and DeBERTa (He et al., 2021) models (base and large). Our evaluation shows that models trained with the masked-regression objective generalize better than most feature-based approaches to C-Tests with different gap sizes and placements. Nonetheless, we find that the features remain important as the best performing model is a feature-based tree-boosting model (XGBoost; Chen and Guestrin 2016) that outperforms all pre-trained language models (PLMs). Finally, perform extensive ablation experiments using this model and find that all feature categories substantially contribute towards the overall performance; indicating the importance of models that are grounded on proper linguistic theory. All works also make substantial code and data contributions—most notably two executable `.jar` files and bash scripts to run the full feature extraction pipeline and 25 fine-tuned difficulty prediction models—published under open source licenses.

Chapter 3

Constrained C-Test Generation

So far, we have discussed the importance of second language acquisition, finding that C-Tests are a well-suited means to practice newly learned vocabulary together with their inflection. We have further discussed different theories to quantify C-Test difficulty and surveyed existing methods for C-Test difficulty prediction. Having established appropriate foundations, we now motivate the need for C-Test generation strategies that deviate from the commonly-used, static one. We then closer inspect contemporary LLMs, finding multiple reasons why they are not well-suited for C-Test generation, and finally, discuss two approaches that we have developed over the course of this thesis.

3.1 Motivation

Despite their advantages over cloze tests and other gap filling formats, a major shortcoming of C-Tests is the static generation strategy that turns every second half (rounded up) of every second word into a gap. Using this strategy as the primary means to automatically generate C-Tests leads to two issues that need to be addressed. First, there still remains a chance that the resulting gaps are ambiguous. This is especially the case in compound words such as "grandmother" (cf. Table 3.1), where the fixed gap size of the static generation strategy results in the hint "grand", leading to an ambiguous gap. Second, the fixed placement of gaps results in C-Tests that are solely dependent on the input text. In other words, the difficulty of a C-Test can only be controlled by the choice of input text, as each input text is turned into exactly one C-Test with a specific difficulty. Ultimately, this ties each input text to a specific difficulty, putting an implicit restriction on the number of input texts that are well-suited for learners of a specific proficiency. In cases where it would be desirable to use a specific input text (for instance, one that was discussed in class), using the static generation strategy is likely to result in a C-Test that needs to be adapted post-generation. However, manually adjusting C-Tests towards a specific difficulty is very challenging due to interdependencies between gaps (Beinborn, 2016); i.e., how changing the size or placement of a single gap propagates to the surrounding gaps, changing their difficulty as well.

Although early works suggest that there may be various benefits connected to deviating from the static generation strategy, this line of research has not received much attention. For instance, Cleary (1988) propose X-Tests that place the gap at the beginning of a word instead of at the end and achieve a substantial increase in difficulty as well as discriminative power (see, e.g., Lord and Novick 1968). Kamimoto (1993) and Farhady and Jamali (2006) come to a similar conclusion by simply varying the gap size (although they do so by hand). Finally, Laufer and Nation (1999) show how manually modifying the gap placement allows teachers to generate exercises of a specific difficulty.

Developing automated generation strategies that deviate from the static generation strategy thus has the potential to substantially ease the work of teachers, allowing them to tailor C-Tests towards their specific needs. Moreover, automatically generating C-Tests would further increase their potential to be used more efficiently in self-directed learning scenarios.

Word	Hint	Gap	Difficulty
grandmother	grand	mother	high
grandmother	grandm	other	low

Table 3.1: Although C-Tests greatly reduce the occurrence of ambiguous gaps, there is no guarantee that they will not occur. For instance, removing the second half of the word "grandmother" results in the hint "grand"—an ambiguous gap with "mother" and "father" as possible solutions. This problem could be easily solved by deviating from the static generation strategy and showing "grandm" as a hint instead.

3.2 Requirements

Our goal is to develop novel C-Test generation strategies that are useful for teachers but can also be utilized in self-directed learning scenarios. For this, we identify two key requirements which novel strategies need to satisfy in order to generate C-Tests that are useful for both scenarios.

First, novel generation strategies should inherently be able to generate C-Tests of a specific *target difficulty* τ. This is important to ensure that learning progresses optimally by allowing teachers to generate exercises that lie within a learner's zone of proximal development (Vygotsky, 1978). As shown in Figure 3.1, generating exercises that are too easy may lead to boredom, while exercises that are too hard may lead to frustration (indicated by the red zones). To keep the learner motivated, it is hence important to generate exercises that are neither of both (i.e., lie within the green zone).

Second, generation strategies should be capable of complying to constraints that are requested by teachers or learners. This would ensure that the generated C-Tests fit their curriculum, for instance, by explicitly turning newly learned vocabulary into gaps or reducing (increasing) the number or size of gaps depending on the learner's proficiency. While generation strategies that fulfil this requirement would greatly empower teachers and learners, the mathematical guarantees they need to provide pose a substantial challenge. This is also the main reason why contemporary state-of-the-art LLMs are not well suited for C-Test generation: they still struggle with hallucination and alignment (Russell, 2019). Figure 3.2 shows an example where the model's output is misaligned with the user's instruction.[1] It is apparent that the model is incapable of identifying and fulfilling all hard constraints posed by the user; namely, that the generated exercise must contain three sentences and ten gaps. Consequently, a teacher would still have to manually place additional gaps in the exercise to achieve their goal.

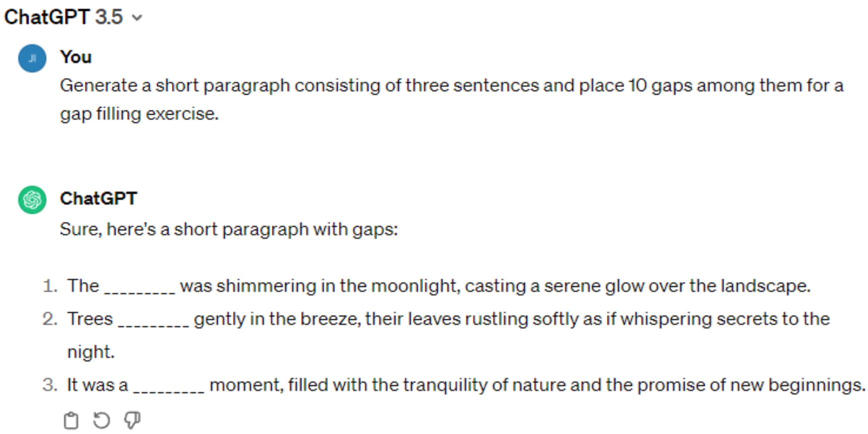


Figure 3.2: GPT-3.5 failing to comply with all instructions provided by the user.

In this thesis, we thus develop novel C-Test generation strategies that adhere to both requirements. They are inherently capable of generating C-Tests of varying difficulty and at the same time, are mathematically guaranteed to fulfil all constraints posed by teachers and learners.

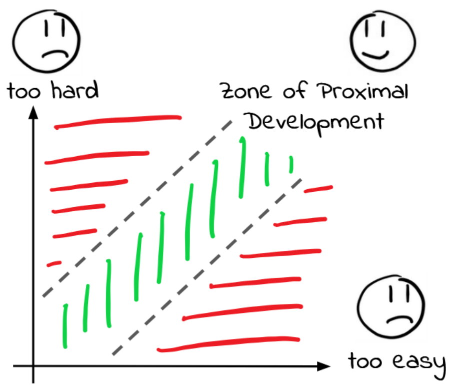


Figure 3.1: The zone of proximal development (green). Generated C-Tests should be neither too hard nor too easy (red) to keep learner motivation high.

3.3 C-Test Generation

Although a large number of works explore the task of exercise generation, many focus on exercise types different from C-Tests. Especially the increasing capability of generative models has instead stoked various works to explore the task of question generation for reading comprehension exercises (Zou et al., 2022; Perkoff et al., 2023; Xiao et al., 2023). However, not many are concerned with assessing the difficulty—let alone controlling the difficulty. Moreover, many works conduct evaluation with NLG-focused metrics such as ROUGE (Lin, 2004) or n-gram overlaps (Rathod et al., 2022) that make it difficult to assess educational value. For reading comprehension exercises,

[1] Date of conversation with ChatGPT (GPT-3.5): 23.02.2024
URL: https://chat.openai.com/share/2ca36950-30d1-4991-ad73-a32d076ff98a

we find that only Uto et al. (2023) and Laverghetta Jr. and Licato (2023) consider exercise difficulty and actual human evaluation. Fortunately, we find a different trend in the second prominent research area concerning exercise generation; namely, cloze tests. Here, most works focus on selecting suited distractors—in a way that allows teachers to explicitly control the difficulty of the generated cloze tests (Yeung et al., 2019; Gao et al., 2020; Panda et al., 2022) or to practice a specific competence such as grammar (Yoshimi et al., 2023). Finally, Felice et al. (2022) propose a new method to fine-tune ELECTRA (Clark et al., 2020) for generating open cloze tests and find that their model performs mostly well but also suffers from structural problems. For C-Tests, we are the first to develop automated generation strategies with a controllable difficulty and thus, provide a task formalization in the following.

3.3.1 Task Definition

Following our definitions from §6.3 and §7.3, we define the task of C-Test generation as consisting of the following inputs:

$\mathcal{T}$: the input text used as the basis.

τ : the target difficulty of our C-Test (i.e., the true score).

m : the number of gaps to be placed in the C-Test.

We further define the set $\mathcal{G} \subset \mathcal{T}$ that includes all words w that can be turned into a gap g with $m < n = |\mathcal{G}|$.[2] Following CTT and Equation (2.8), we define the true score $\tau \in [0, 1]$ as:

$$\tau = \frac{1}{m}\sum_{i=1}^{m} \texttt{error}(g_i), \tag{3.1}$$

where g_i denotes the i-th gap of the C-Test. The function $\texttt{error}(\cdot)$ indicates if the i-th gap was filled-out correctly and returns a binary value (0 for correct and 1 for incorrect). Consequently, smaller values of τ relate to easier, and larger values to more difficult C-Tests. As the actual $\texttt{error}(\cdot)$ function is learner dependent and not known during generation, we approximate it using a gap difficulty prediction model $f_\theta : \mathbb{R}^k \mapsto [0, 1]$ with parameters θ that computes the error rate for each gap, represented as a k-dimensional, real number vector $\mathbf{x}$.[3] We can now define the estimated difficulty $\hat{\tau}$ for any selection of m gaps $g \in \mathcal{G}$:

$$\hat{\tau} = \frac{1}{m}\sum_{i=1}^{m} f_\theta(g_i), \tag{3.2}$$

and say that given f_θ, any C-Test that has minimal distance between the estimated and the target difficulties is optimal. Hence, our optimization objective is:

$$\min \quad |\tau - \hat{\tau}|. \tag{3.3}$$

[2]Note, that $\mathcal{G}$ excludes words that contain any numbers or punctuation and have less than two characters; i.e., words that either violate the C-Test creation paradigm by providing no hint or are likely to result in gaps unrelated to language proficiency (e.g., the mention of years such as "1642").

[3]Depending on the used model, each of the k dimensions either present a feature or an embedding dimension. For instance, the XGB model used in Chapter 7 uses $k = 61$ features while the BERT model in the masked-regression setup uses $k = 512$, corresponding to the embedding dimension.

Gap placement. So far, $\hat{\tau}$ only includes gaps that have already been selected. To model the task of optimally placing them across all possible gaps $\mathcal{G}$ with $|\mathcal{G}| = n > m$, we now introduce binary decision variables:

$$\min_{b_i \in \{0,1\}} \quad |\tau - \frac{1}{m} \sum_{i=1}^{n} b_i \, f_\theta(g_i)| \tag{3.4}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} b_i = m, \tag{3.5}$$

where b_i denotes a binary decision variable for a selected gap at the i-th word.[4] The constraint ensures that the resulting C-Test has exactly m gaps.

Gap size. In addition to the placement, each gap can assume a different size. We hence extend our objective with additional binary decision variables $s_{i,j}$ for the gap size where l_i denotes the length of the i-th word with $j \in \{1, \ldots, l_i - 1\}$. Our final model comprising gap placement and gap size is then:

$$\min_{s_{i,j}, b_i \in \{0,1\}} \quad |\tau - \frac{1}{m} \sum_{i=1}^{n} b_i \sum_{j=1}^{l_i - 1} s_{i,j} f_\theta(g_{i,j})| \tag{3.6}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} b_i = m, \tag{3.7}$$

$$\sum_{j=1}^{l_i - 1} s_{i,j} = 1, \tag{3.8}$$

where $g_{i,j}$ denotes a gap at the i-th word with size j, for all words $i \in \{1, \ldots, n\}$. Our binary decision variables for the gap placement (b_i) and size ($s_{i,j}$) have one constraint each to ensure that the resulting C-Test consists of exactly m gaps (Equation (3.7)) and that each gap has exactly one specific gap size (Equation (3.8)).

3.3.2 Methods

Existing works only consider the manual manipulation of C-Test difficulty (Cleary, 1988; Kamimoto, 1993; Laufer and Nation, 1999; Grotjahn, 2006). Automatically generating C-Tests that suit a target difficulty would substantially increase their versatility—allowing any text to be used as an input—but comes with substantial challenges as C-Test difficulty is influenced by various factors (Beinborn, 2016). In this thesis, we devise two fundamentally different C-Test generation methods which we will outline in the following. Chapter 6 introduces two generation strategies that either manipulate the gap size or gap placement of C-Tests (but not both). We will refer to them as *locally optimal* approaches, as the manipulation takes place iteratively. In Chapter 7, we devise an approach based on *constrained optimization* methods that are capable of complying with hard constraints. We will refer to them as *globally optimal* approaches, as they provide rigorous mathematical guarantees for optimization.

[4] Note, that i now ranges from 1 to n, with n denoting the number of all possible gaps and m the target number of gaps.

Locally optimal approaches. A key concern of our work in Chapter 6 is the large number of possible combinations between gap size and placement during generation. Only considering all possible gap placements without varying the gap size already results in:

$$\binom{n}{m} = \frac{n!}{m!(n-m)!} \tag{3.9}$$

C-Tests for placing m gaps across n words. Consequently, simply generating every possible C-Test out of $\mathcal{T}$ and selecting the one with minimal distance to τ is infeasible. Instead, we propose to partially rely upon the static generation strategy and develop two iterative approaches that either manipulate the gap placement (`SEL`) or size (`SIZE`):

`SEL` first estimates the difficulty $\hat{\tau}_i$ of all $n = 40$ candidate gaps $g_i \in \mathcal{G}$ using an SVM trained with the features defined by Beinborn (2016). All gaps are then divided into two sets $\mathcal{G}_{\leq\tau} : \forall\hat{\tau}_i \leq \tau$ and $\mathcal{G}_{>\tau} : \forall\hat{\tau}_i > \tau$; that is, sets that only contain gaps g_i that are easier or harder compared to the target difficulty τ. $\mathcal{G}_{\leq\tau}$ and $\mathcal{G}_{>\tau}$ are then both sorted according to $|\tau - \hat{\tau}_i|$ (i.e., their distance to τ). Finally, we generate the C-Test by selecting the gaps closest to τ from each set in an alternating manner, until the final number of gaps $m = 20$ have been selected.

`SIZE` also starts with estimating the gap difficulty of all default gaps using the same model as in `SEL`. We then utilize two SVMs that we separately train to predict the relative change in difficulty if the gap size is increased or decreased by one character. For training these SVMs, we conduct an ablation study, identifying six features that minimize compute time while maximizing performance. To provide both SVMs with sufficient training data, we utilize synthetic data that was generated by estimating the difficulty of gaps with varying gap sizes using the full feature model. Using these models as our estimators, we then increase or decrease the gap size for each gap until all gaps have a minimal distance to the target difficulty τ.

Note, that both strategies utilize parts of the static generation strategy, but in different manner. `SEL` only varies the gap placement, keeping the gap size static and `SIZE` only varies the gap size with a static gap placement. Although we evaluate additional models and features for both strategies in subsequent work (cf. §7.8, D1.2), we find that the models trained in Chapter 6 perform better.

Globally optimal approaches. One shortcoming of our work in Chapter 6 is the predefined restriction of the generated C-Test in terms of gap size (`SEL`) or gap placement (`SIZE`). This discards a large number of C-Tests as potential solutions. Moreover, iteratively selecting gaps or adjusting their size does not take any global interdependencies between gaps into account, leading to locally optimal solutions. In Chapter 7, we hence propose to tackle C-Test generation as a mixed-integer programming (MIP) problem which results in three advantages over existing methods.[5] First, we can now make use of well-established solving methods that efficiently remove whole sets of unsuitable C-Tests while finding a provably optimal solution (Schrijver, 1986). This allows us to consider all possible C-Tests with a feasible run time (~48.6 seconds). Second, we can directly

[5]For an introduction into MIP, we refer the interested reader to §7.8.A.2.

integrate trained models into the optimization problem, providing an end-to-end solution that achieves global optimality. Third, the use of MIP provides mathematical guarantees that the resulting C-Test adheres to all constraints that are defined by a teacher.

A key contribution of this work is the formalization of our model f_θ that is comprised of features proposed by Beinborn (2016) and Lee et al. (2020). This results in 61-dimensional vector $\mathbf{x}$ for a single gap $g_{i,j}$, where each dimension k relates to a specific feature. Overall, we identify three types of features. First, 51 features that are not affected by a change in gap size $s_{i,j}$ or gap placement b_i. We model these as equality constraints:

$$g_{i,j,k} = c_{i,j,k}, \tag{3.10}$$

where $c_{i,j,k}$ is the k-th pre-computed feature value for gap $g_{i,j,k}$. Second, six features that only change according to the gap size $s_{i,j}$ which we model as an additional constraint:

$$g_{i,j,k} = \mathbf{s}_i \cdot \mathrm{C}_{i,k}, \tag{3.11}$$

where $\mathbf{s}_i \in \mathbb{B}^{l_i - 1}$ is the 1-hot vector (of length $l_i - 1$) representing the gap size decision variables (with 1 being at the j-th position) for gap g_i. The matrix $\mathrm{C}_{i,k} \in \mathbb{R}^{l_i - 1 \times k}$ represents all possible values $k \in \mathcal{U}$ can take for all possible gap sizes $j \in \{1, \ldots, l_i - 1\}$ at gap g_i with $\mathcal{U} = \{49, 50, 56, 57, 58, 59\}$; i.e., all our features that depend on the gap size. Third, four features that depend on the gap placement. In contrast to the gap size features, all placement features depend on the placement of the other gaps. We thus need to model these dependencies into our constraints:

$$g_{i,j,51} = \max(\mathbf{b} \cdot \mathrm{V}), \tag{3.12}$$

$$g_{i,j,52} = \sum_{h \in \mathcal{S}_i} b_h, \tag{3.13}$$

$$g_{i,j,53} = \sum_{h=1}^{i-1} b_h, \tag{3.14}$$

$$g_{i,j,54} = \sum_{h \in \mathcal{S}_i, h < i} b_h, \tag{3.15}$$

for all $i, j \in \{1, \ldots, n\}$ where $\mathcal{S}_i$ denotes the set of all words in the sentence containing g_i. The vector $\mathbf{b}$ denotes all placement decision variables b_i and V the $n \times n$ matrix of binary values $v_{i,j}$ with:

$$v_{i,j} = \begin{cases} 1, & \text{if } w_i = w_j, \\ 0, & \text{otherwise,} \end{cases}$$

for all $i, j \in \{1, \ldots, n\}$ where $i \neq j$. A full list of features is provided in §7.8.B.1.

3.4 Contributions

We are the first to develop methods that automatically generate C-Tests of varying difficulty from a single input text. This is a substantial contribution which allows teachers and self-directed learners to choose the input text based on their curriculum or personal interest and not based on the difficulty of the resulting C-Test (as with the static generation strategy). In total, we devise three novel generation strategies in two works. In Chapter 6, we propose two locally optimal generation strategies that either modify the gap size or placement. To make these strategies computationally feasible, we conduct ablation studies and identify a small subset of features that are used to train efficient gap difficulty prediction models. Our user study shows that both generation strategies succeed in controlling the difficulty of the resulting C-Test. In our evaluation with synthetic data we identify the restriction to either modify the gap size or placement (but not both) as one of the key limitations of both locally optimal generation strategies. In Chapter 7, we tackle this limitation by devising a globally optimal generation strategy that is capable of modifying gap size and placement at once. We further find that contemporary LLMs such as GPT-4 (OpenAI, 2023) struggle to fulfill hard constraints (e.g., fail to always generate the correct number of gaps) which motivates us to utilize constrained optimization methods. The resulting mixed-integer programming formulation of our novel generation strategy provides mathematical guarantees to adhere to hard constraints posed by teachers. Our user study shows that our proposed method significantly outperforms two of the baseline strategies (based on gap placement and GPT-4); and performs on-par with the third (based on gap size). Finally, we devise and evaluate additional formulations of the optimization objective, substantially reducing the run time of the solver from 22.5 to 3.1 seconds. Both works also make substantial code and data contributions—most notably 48 C-Tests consisting of 20 gaps each and annotated with their respective error rate—published under open source licenses.

Chapter 4

Adaptive C-Test Selection

In the previous chapter, we have introduced methods that allow us to automatically generate C-Tests that suit a learner's proficiency. In the context of exercise generation, we have identified two requirements that are important. First, methods should be capable of generating exercises that adhere to a target difficulty and second, they should adhere to constraints posed by teachers. We have discussed how both requirements are necessary to alleviate the work of teachers and to empower self-directed learning. Finally, our evaluation studies showed that all developed methods are advantageous over the static generation strategy. This summarizes the first major contribution of this thesis, which paves the way towards building more open educational resources through automated generation methods. The second major contribution focuses on two aspects that have received less attention in this thesis so far. First, *adapting* models to an individual learner or a specific group of learners and second, *selecting* the most appropriate C-Test out of a pool of already tried and tested C-Tests.

4.1 Motivation

So far, our C-Test generation strategies rely on models that assess the general C-Test difficulty regardless of the individual learner; or in other words, utilize the mean error rate E from all learner responses as shown in Equation (2.8). This may be fine in scenarios where a teacher (or a system) has enough knowledge to assess a learner's proficiency, allowing them to consider it during exercise selection or generation. But this is not always the case, especially in self-directed learning scenarios. Imagine a learner who has already completed an introductory course and wants to keep practicing on their own using an intelligent tutoring system (ITS; see, e.g., Murray 1999). Without asking the learner about their previous knowledge, the system has no way besides random selection to suggest exercises. This may be fine in the beginning, however, continuously doing so would substantially hurt the learning process and demotivate learners due to exercises that are too easy or too difficult. To provide better suggestions, the system has to consider the learner's feedback (e.g., the errors they made on a previously suggested exercise) and adapt its suggestions towards it. It is important to note that this problem does not only arise in the beginning—as learner proficiency increases, the system has to further adapt its suggestions to keep the learner engaged (Illeris, 2003). This brings us to *interactive machine learning* (IML; Fails and Olsen 2003) where the model is iteratively updated with incoming feedback from a human.

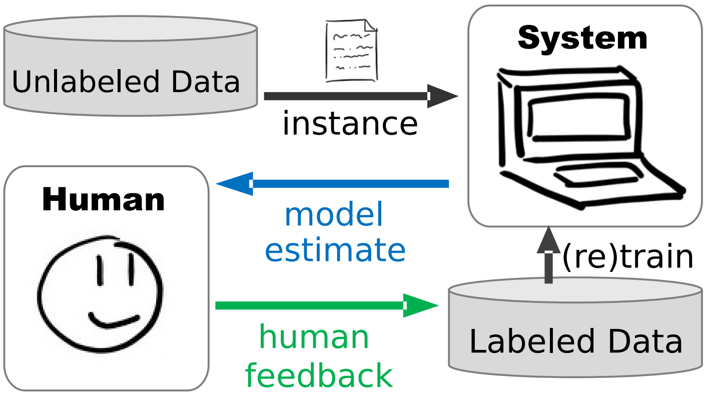


Figure 4.1: Interactive machine learning (IML). In contrast to classical machine learning where a model is only trained once, IML considers a continuous learning scenario where a model is updated with newly provided human feedback.

Figure 4.1 shows the typical IML cycle where a system iteratively updates its internal machine learning model with incoming human feedback. One such example are recommender systems (cf., Aggarwal 2016 for an introduction) where a user is presented with a list of items to choose from (ranked according to the model's estimates). By choosing the item that best fits their preference, the user provides feedback which is then used to retrain the model. Over the years, IML has been researched across a wide range of scenarios that involve human interaction and has benefited from research in related areas such as continual learning, preference learning, and reinforcement learning (Mosqueira-Rey et al., 2023). A key challenge when deploying IML in educational scenarios is the inherent competency of the human. In recommender systems, users are typically aware of their likings to make choices that suit their needs. In contrast, the goal in educational scenarios is to provide learners with exercises that lead to an optimal learning process. Asking learners to assess the usefulness of an exercise themselves (e.g., by choosing from a list of exercises) is likely to result in suboptimal choices as they are not experts (i.e., teachers). A more reliable way to obtain feedback is through a learner's performance on a selected exercise. This requires us to devise *data efficient* methods, as learners cannot be asked to solve an arbitrary number of exercises just for adapting the model towards their needs.

4.2 Data Efficiency

To improve data efficiency, we can either use fewer training instances, or make better use of available instances. Despite recent works that emphasize the importance of data quality (Paullada et al., 2021; Lee et al., 2022b; Kreutzer et al., 2022), data work often remains a less popular research area (Sambasivan et al., 2021). Overall, we identify three lines of research that aim to improve data efficiency in different ways (Treviso et al., 2023): *filtering*, *curriculum learning*, and *active learning.*[1]

4.2.1 Filtering

Improving *data quality* by filtering out low-quality instances can boost performance while reducing training costs during pre-training and fine-tuning.[2] Especially for limited compute budgets, a careful selection of instances is important to achieve optimal performance (Hoffmann et al., 2022). Filtering approaches such as deduplication have been shown to be very effective for pre-training (Lee et al., 2022b). Finally, filtering can also mean removing instances that do not contribute much to model training. One such example is the SNLI dataset (Bowman et al., 2015), for which Mishra and Sachdeva (2020) identify a subset of ∼2% that results in a similar fine-tuning performances as to using the full corpus.

4.2.2 Curriculum Learning

Whereas filtering relates to a static removal of data, curriculum learning follows a more dynamic process by identifying data orderings that reduce the number of training steps to achieve a target performance (Elman, 1993; Bengio et al., 2009). Key challenges in curriculum learning are quantifying the instance difficulty (Swayamdipta et al., 2020; Baldock et al., 2021; Ethayarajh et al., 2022) and finding a good *pacing*, i.e., when to progress to more difficult instances to avoid wasting compute on "easy" instances (Kumar et al., 2010). Despite various works that devise new difficulty estimates and pacing strategies (Kocmi and Bojar, 2017; Graves et al., 2017; Platanios et al., 2019) selecting the best one for a specific task remains difficult (Dodge et al., 2020).

4.2.3 Active Learning

Finding that instances and their ordering contribute differently towards model performance raises the question why one should annotate all instances—including those with little or no contribution—in the first place. This brings us to *active learning*, the third and central line of research for this chapter. Active learning aims to effectively reduce annotation cost and the amount of necessary training data by only labeling instances that contribute most towards improving a model's performance (Lewis and Gale, 1994; Settles, 2012). This is by no means easy, as the usefulness of instances needs to be estimated before annotation; i.e., without knowing the ground truth.[3] Many works resort

[1] Note, that the term active learning also exists as a teaching paradigm where the goal is to actively involve learners (Bonwell et al., 1991). In this thesis, we use this term to exclusively refer to the machine learning paradigm.

[2] Here, we focus on unlabeled data or datasets that have already been cleaned. For approaches that tackle erroneously labeled instances, we refer to Klie et al. (2024).

[3] This is a key difference to curriculum learning that relies upon knowing the ground truth (which is not necessarily the same as a label). One example is self-supervised learning that utilizes the next word as a ground truth (hence, requiring no label) and for which curriculum learning has been applied (Nagatsuka et al., 2021).

to estimating the model *uncertainty*—assuming that labeling instances with the highest uncertainty is most helpful (Lewis and Gale, 1994; Tang et al., 2002; Gal et al., 2017; Yuan et al., 2020) and instance *representativeness*—to maximize diversity of sampled instances while avoiding outliers (Bodó et al., 2011; Sener and Savarese, 2018); or a combination of both criteria (Kirsch et al., 2019; Ash et al., 2020; Margatina et al., 2021).

Advantages. Following the compelling promise of efficient model training and reduced annotation costs, works have studied and successfully employed active learning in various areas such as MT (Liu et al., 2018), entity linking (Klie et al., 2020), and coreference resolution (Li et al., 2020; Yuan et al., 2022). Active learning has also been shown to be helpful across different training paradigms and model architectures (Zhang et al., 2022). For instance, Ein-Dor et al. (2020) study various active learning strategies across different text classification tasks using BERT (Devlin et al., 2019). Su et al. (2023) even show how active learning can improve performance for in-context learning (ICL; Radford et al. 2019). Especially in domains such as education and language learning that are constantly challenged by privacy concerns and proprietary datasets (Poesio et al., 2013; Lyding et al., 2022) active learning can hence provide a compelling solution.

Shortcomings. Despite all advantages, various issues remain such as the correct choice of model, the hyperparameters, and stopping criterion (Lowell et al., 2019; Margatina and Aletras, 2023). Works have further found that active learning can be prone to selection biases and may favor outliers (Cortes et al., 2008; Karamcheti et al., 2021; Snijders et al., 2023); or that it may even involuntarily increase the annotation cost by primarily selecting instances that are "difficult" to annotate (Settles et al., 2008; Lee et al., 2022a). All these issues can be addressed with the development of new methods that are outlier-aware (Kirsch et al., 2021) and by adapting existing methods to new models (Margatina et al., 2022). However, a fundamental challenge that prevents the deployment of active learning in interactive scenarios is the assumption of an omnipresent and omniscient *oracle* (i.e., human annotator) (Kottke et al., 2017; Settles, 2009). As we will discuss in the next section, the main issue here is not assuming that the oracle is omnipresent and omniscient. Especially the latter has been addressed in various efforts to adapt active learning to crowdsourcing by modelling noisy labels (Yan et al., 2016; Zhang and Chaudhuri, 2015; Lin et al., 2016; Baumler et al., 2023). The main issue is reducing humans to the role of a mere labeling oracle (Amershi et al., 2014).

4.3 Active Learning for Interactive Scenarios

In annotation scenarios, humans assume the role of an labeler with an explicit incentive to label instances. Moreover, they are intrinsically motivated to annotate as they receive some kind of reward (e.g., money or data). This is completely different in interactive scenarios where humans are users of a system with an incentive that is not labeling. Take for instance recommender systems, where the user's incentive is to receive the best possible recommendation of a product. Although the system may be adapted interactively with a user's feedback, its primary goal remains to make suggestions that satisfy the user's goal. Same goes for the educational domain. For instance, an ITS should aim to select the exercises that maximize a learner's progress (Sottilare et al., 2017). This however leads to the issue that the selected instances may not be the ones that substantially improve the system and moreover, it may even be reasonable to select instances where a system is highly certain (i.e., the exact opposite of active learning). Consequently, it is highly likely that the user's feedback is not very useful which slows (or may even halt) the system's improvement. In this thesis, we tackle this issue by conjoining active learning and interactive scenarios; devising methods that jointly optimize model and user objective.

4.3.1 Formalization

We first formalize both objectives in the context of learner adaptive exercise selection. Let $\mathcal{U}$ by the pool of unlabeled exercises. At each iteration i, we select an exercise $x_i \in \mathcal{U}$ using a model $f_{\theta_{i-1}}$ and query it to a learner v_{i-1}. The learner v_{i-1} then solves the exercise, improving their proficiency (resulting in v_i for the next iteration) and providing a respective label y_i (i.e., their performance on exercise x_i). The resulting exercise-label pair (x_i, y_i) is added to the set of labeled exercises $\mathcal{L}$ and x_i is removed from $\mathcal{U}$. Finally, we train (or update) our model f_{θ_i} using $\mathcal{L}$.

Model objective. Following the active learning paradigm, the model's goal is to select exercises that would maximize its predictive performance if labeled. We thus define model objective s_{model} as:

$$s_{\text{model}}(\mathcal{U}, f_{\theta_{i-1}}, v_{i-1}) = \arg\max_{x \in \mathcal{U}} U(x, f_{\theta_{i-1}}), \tag{4.1}$$

where $U\colon (x, f_{\theta_{i-1}}) \mapsto [0,1]$ is a function that estimates the usefulness of x for improving $f_{\theta_{i-1}}$; for instance, the maximum entropy (Shannon, 1948) in the case of multi-class uncertainty sampling (Lewis and Gale, 1994).

User objective. In contrast to the model objective, the learner's goal is to achieve the best possible learning process; i.e., to receive instances that maximize their language proficiency. We therefore define the user objective s_{user} as:

$$s_{\text{user}}(\mathcal{U}, f_{\theta_{i-1}}, v_{i-1}) = \arg\max_{x \in \mathcal{U}} A(x, v_{i-1}), \tag{4.2}$$

where $A\colon (x, v_{i-1}) \mapsto [0,1]$ returns the degree of appropriateness of instance x for the learner v_{i-1}. We consider an exercise appropriate if it is neither too easy nor too difficult, as this maximizes the learner's proficiency. $A(\cdot)$ can be quantified in various ways, for instance via CTT (cf. §2.3) by measuring the error between the predicted label and the learner's demand (Lee et al., 2020).

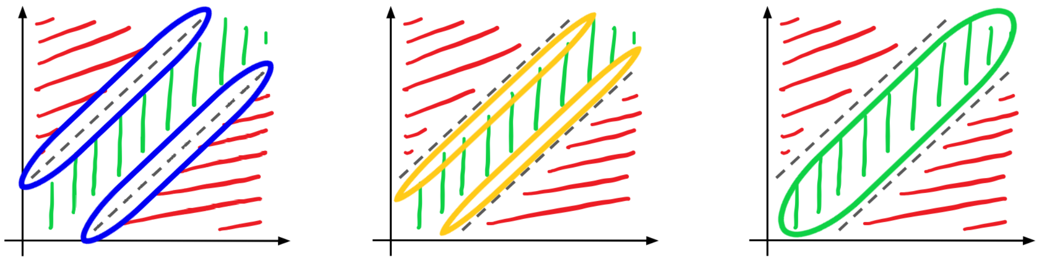


Figure 4.2: Sampling objectives visualized within the zone of proximal development. Green and red regions indicate exercises that are desirable or to be avoided, respectively. The model objective (left; in blue) focuses on exercises from the border region that presumably help the model to learn the borders well. In contrast, the user objective (right; in green) requires exercises that suit the learner well. In the joint objective (middle; in gold), we propose to sample exercises that benefit both, model and user objective.

4.3.2 Jointly Optimizing Model and User Objectives

We have defined model and user objectives in a flexible manner which allows us to utilize any arbitrary function for the usefulness $U(\cdot)$ and appropriateness $A(\cdot)$, respectively. Although both objectives seem counteracting at first glance, the zone of proximal development provides us with an intuitive explanation on how they are jointly optimized. Figure 4.2 visualizes all sampling objectives in the zone of proximal development (Vygotsky, 1978) with red and green indicating regions of exercises that are ill-suited and well-suited, respectively. The goal of our model is to classify exercises into ill-suited and well-suited ones; i.e., to learn the dashed borders that separate the red and green regions. Consequently, sampling exercises from the border regions is most beneficial for the model (left; in blue). In contrast, the goal of the learner is to receive exercises from within their zone of proximal development; i.e., the green region (right; in green). In Chapter 9, we show that it is possible to optimize both by sampling exercises from the intersecting regions (middle; in gold).

4.3.3 Adaptive Exercise Selection

Finally, we discuss existing works that investigate methods for adaptive exercise selection (or generation). Here, we focus on methods that are adaptive themselves; i.e., that are capable of learning from a learner's feedback. This is in stark contrast to systems that are capable of providing learner adaptive suggestions but only based on a pre-defined set of rules (or models); in other words, without any capability of automated adaptation (Baker, 2016; Essa et al., 2023; Kaur et al., 2023). Although such systems may somewhat alleviate existing bottlenecks in second language acquisition and have been developed for various exercise types (Haring et al., 2021; Heck and Meurers, 2022; Chan et al., 2022; Bitew et al., 2023)—adapting them to specific learners or use cases requires substantial work.

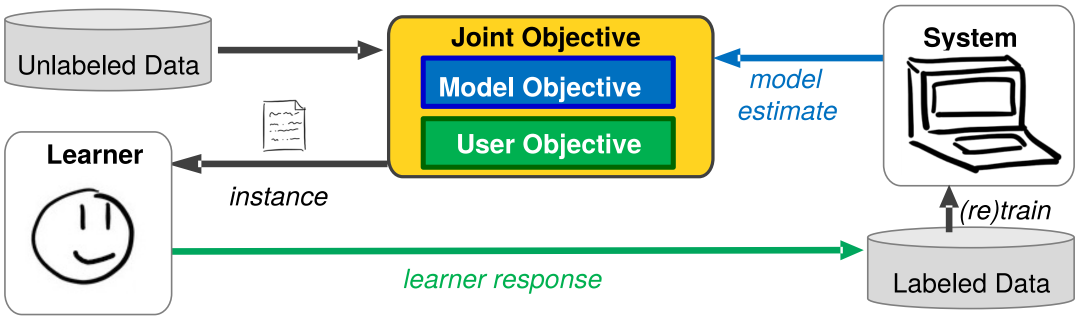


Figure 4.3: Efficient, adaptive exercise selection. We conjoin active learning (blue) and leaner goals (green) to jointly optimize these seemingly counteracting goals (gold). Figure adapted from Lee et al. (2020).

Devising methods that automatically adapt themselves to suit a leaner's need is hence essential to meet the increasing demand on language learning applications.

Apart from some early works that utilize interactively trained SVMs for adaptive test selection (Hoshino et al., 2010) or partially observable Markov decision processes (POMDPs) to suggest actions in ITS (Chi et al., 2010; Rafferty et al., 2016); adaptive exercise selection has only gained more attention recently (Truong, 2016). For instance, Heck and Meurers (2023) investigate the use of learning analytics (Siemens and Long, 2011) to dynamically generate personalized exercises. Cui and Sachan (2023) propose to a system that keeps track of a learner's progress using an LSTM (Hochreiter and Schmidhuber, 1997) for deep knowledge tracing (Corbett and Anderson, 1994; Piech et al., 2015) and a BART (Lewis et al., 2020) model to generate vocabulary training exercises (Settles et al., 2018).

C-Test selection. With respect to C-Tests, we identify three works that propose adaptive selection methods. Settles et al. (2020) propose an IRT-based method to adaptively select various kinds of exercises; comprising audio and textual yes or no questions, C-Tests, dictation exercises, and speech exercises. They train two regression models to predict exercise difficulty on a CEFR-scale (Council of Europe, 2001) and find considerable performance in terms of ranking correlation. McCarthy et al. (2021) improve this model even further by introducing additional BERT features to address the cold start problem (Schein et al., 2002).[4] Most notable in both works is the large amount of exercises that are used for evaluation, however, due to proprietary restrictions, neither code nor data is publicly available. Finally, we propose an adaptive selection method using CTT-based, simulated learners in Chapter 9. Our experiments show that jointly optimizing model and user objective leads to exercises that optimally train the model

[4]The cold start problem comes from the area of recommender systems and refers to the scenario when no (or only little) data is available for "warming up" (i.e., training) a recommender model. Utilizing heuristics or pre-trained models until a sufficient amount of data is available, is one way to address this problem.

(compared to uncertainty sampling) and at the same time, fits the learner's proficiency. Figure 4.3 shows an overview of our proposed system.

4.4 Contributions

We define adaptive C-Test selection as a task complementary to C-Test generation which can alleviate the work of teachers and empower self-directed learning. We then discuss the benefits of IML for educational scenarios and identify data efficiency as a key property for a successful deployment. Our survey across different fields related to data efficiency allows us to single out active learning as the most promising research direction as it substantially increases data efficiency and improves model training. We then propose an adaptation of active learning to IML scenarios, by separately formalizing the model objective, i.e., selecting instances that optimally train the model once labeled, and the user objective, i.e., selecting instances that lead to an optimal learning process. This marks the second key contribution of this thesis. Using this formalization, we then devise a sampling strategy in Chapter 9 that selects C-Tests which jointly optimize both objectives; resulting in exercises that efficiently inform the difficulty prediction model about changes in a learner's proficiency while ensuring that they are neither too easy nor too difficult. Our experiments with simulated learners show that our joint optimization approach yields significant improvements over optimizing each objective individually, indicating that both objectives need to be considered in IML scenarios. We further make substantial code and data contributions—most notably five different learner models and the implementation of five different sampling strategies as well as four different learner behaviors—published under open source licenses.

Chapter 5

Beyond CALL

In the first part of this thesis, we devised novel C-Test generation strategies that are mathematically guaranteed to uphold any constraint that may be posed by teachers. In the second part, we then investigated adaptive C-Test selection methods and found that sampling C-Tests according to a joint optimization strategy benefits both, the learner and the model. Especially the second contribution was made possible by shifting the focus away from a solely model-centered perspective and more towards the human; effectively operationalizing active learning for human-centered IML scenarios. In this chapter, we will show how research areas beyond CALL can equally benefit from focusing more on the human's perspective. More specifically, we will outline two use cases in data acquisition utilizing our insights from previous chapters. In our first use case, we will show how we can utilize interactive data collection to alleviate the work of domain experts and at the same time, collect highly domain specific dialogue data. In our second use case, we will show how learning curricula (Kelly, 1977) can be used in data annotation studies to implicitly train annotators and significantly reduce the annotation time.

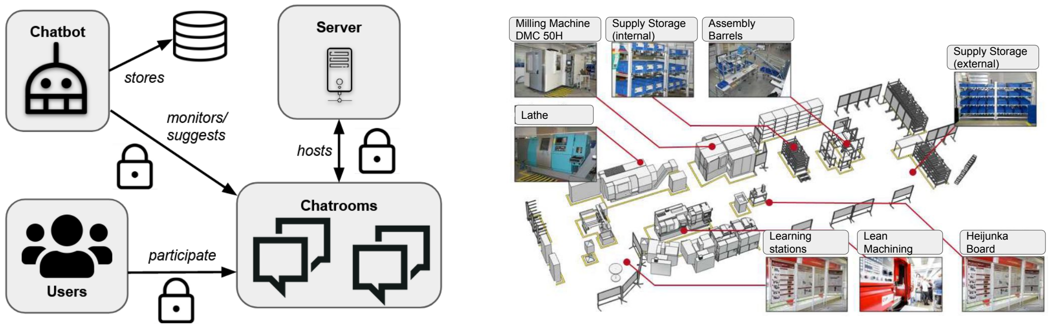
5.1 Motivation

The lack of high quality training data and other open resources which we discussed in Chapter 4 is not only an issue in educational research, but one that concerns the whole AI research community. Although early works already identify and tackle data acquisition as one of the key bottlenecks (Olson and Rueter, 1987; Cullen and Bryman, 1988), data collection still remains challenging (Sambasivan et al., 2021). Especially the complexity of annotation studies—that often involve multiple steps which can vastly change depending on the underlying task, the annotators, and many other factors—makes data acquisition difficult and error-prone. Unfortunately, alleviating the work of (human) annotators is an aspect that receives less attention, even though annotators are the backbone of each annotation study (Klie et al., 2024). Instead, various works propose to reduce annotation costs by recruiting non-expert annotators such as crowd workers (Snow et al., 2008) or by (semi-) automatically generating data (see, e.g., Bañón et al. 2020)—often at the expense of annotation quality (Paullada et al., 2021; Kreutzer et al., 2022). Besides lowering annotation quality, the quest for cheap data has also raised ethical issues (Shmueli et al., 2021; Kummerfeld, 2021) and even allured crowd workers to utilize LLMs for annotation (Veselovsky et al., 2023). This is a concerning development especially in the face of *model collapse*; i.e., the forgetting caused by repeatedly training on automatically generated (i.e., synthetic) data (Shumailov et al., 2024). We propose that one sustainable way to combat this trend is to make manual (expert) annotations more feasible—not by simply reducing the payment but by easing the annotation process using our insights from IML and CALL.

5.2 Interactive Data Collection

The lack of domain experts is one of the key bottlenecks that make data acquisition in many areas difficult or very expensive. Nonetheless, building systems based on expert knowledge can substantially ease the work of the very same experts, especially in areas such as medicine where they have to face many stressful situations (Hummelsberger et al., 2023). To avoid burdening experts with additional data collection work, one solution is to implicitly collect data by integrating data collection into systems that are already used by experts in their daily work. This shares some similarity to *games with a purpose* (Von Ahn, 2006) which have been successfully deployed across various data collection efforts (Chamberlain et al., 2013; Madge et al., 2019; Kicikoglu et al., 2020). However, whereas games with a purpose are built in a bottom-up manner starting from a specific annotation task, this is not possible in scenarios with already existing workflows. Instead, expert scenarios require a top-down approach that considers the full scope of their work and then identifies individual opportunities where systems can provide assistance. Such a human-centered approach guarantees that experts who contribute the data actually benefit from the system, resulting in usable AI (Xu, 2019).

Task-oriented dialogues in a factory environment. We demonstrate the effectiveness of a human-centered, interactive data collection system in a use case with a highly specific domain, namely, task-oriented dialogues that occur between workers in a factory. Task-oriented dialogue processing is a use case where data is scarce and tedious to annotate (Razumovskaia et al., 2022). Collecting dialogue data is even more difficult in languages other than English and for very task-specific application domains where only a small number of experts are sufficiently qualified to be involved (Sambasivan et al., 2021). In such scenarios, we cannot ask crowd workers (Crowston, 2012) or deploy expert annotation tasks with traditional annotation tools such as INCEpTION (Klie et al., 2018), because that would require a substantial amount of time to annotate. To alleviate the need for expert annotations in such scenarios, we propose a system that is capable of collecting and annotating data on the fly. We showcase such a system (called TexPrax) that further assists experts their work with a separate dashboard. Interactively collecting data with TexPrax leads to four advantages over common annotation tools and crowdsourcing. First, the users are the very domain experts that hold qualified conversations in the target-domain. This allows us to directly collect the dialogue data, instead of having to generate it semi-automatically or asking crowd workers who can only provide limited expertise (Raghu et al., 2021). Second, employees have an immediate benefit from annotating and improving the recommendation model that is integrated in the dashboard. Third, they have full control over their own data which saves time for NLP practitioners as it alleviates research data management. Finally, the use of an end-to-end encryption protocol ensures that only parties selected by the employees will have access to the data, even if the server is breached.



TexPrax system   Factory environment

Figure 5.1: Left: An overview of the TexPrax messaging system that is composed of three main components; a chatbot that collects data, a messaging application (client-side) that allows users to chat with each other, and a server that collects and manages the data. Right: The data is then used to collect information about a factory environment where workers have to document errors that occur during work. An automated documentation of these errors alleviates existing workflows of the employees. Figures adapted from Stangier et al. (2022).

Figure 5.1 provides an overview of the TexPrax system (left) and the setup (right): several workers in a factory environment consisting of multiple assembly lines. Generally, errors that occur during work need to be documented within a dashboard manually (Müller et al., 2021). Using TexPrax in this setting has a two-fold benefit. First, it accelerates troubleshooting via a save and secure communication channel. Second, it automatically identifies and documents occurring *problems*, their *cause*, and their *solution*. The chatbot we implement further assists workers in their annotation by allowing them to provide corrections, which we showcase in three separate data collection studies.

5.3 Annotator Training

Annotator training is a reliable way to improve data quality with benefits for experts and crowd workers (Bayerl and Paul, 2011). Although it can substantially improve inter-annotator agreement, a recent survey by Klie et al. (2024) finds that annotators received training in only 18% of the cases and that almost all of them (except one) concerned crowd workers. One reason for this may be that annotator training is tedious to implement, especially for annotation studies that involve a complex annotation scheme. For instance, in their effort to collect high quality argumentative data, Stab and Gurevych (2014) undergo several rounds of training with their annotators, having to readjust their annotation scheme and guideline multiple times. To ease annotator trainings, Tauchmann et al. (2020) propose to systematically split the data into multiple batches of increasing complexity. After each batch, the annotators are evaluated and low-performing annotators are asked to either complete additional trainings, or filtered out. While this effectively

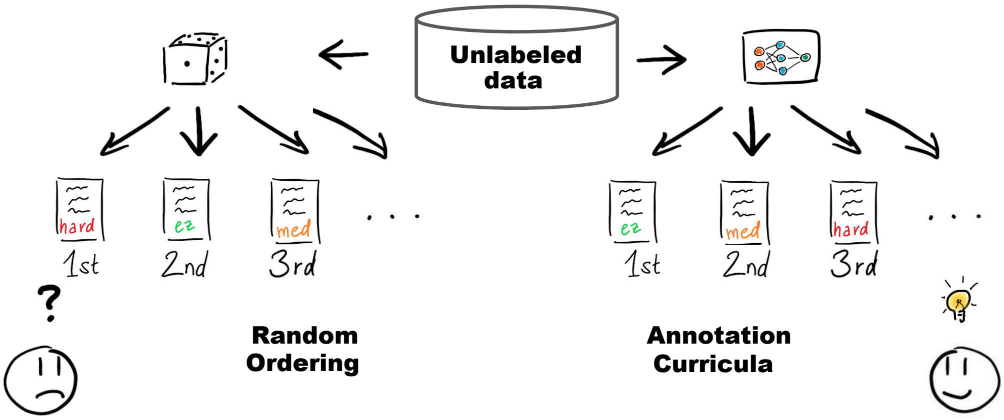


Figure 5.2: Annotation curricula in a nutshell. We show that annotators who receive instances for annotation according to a proper curriculum (right) require significantly less time, compared to a random ordering (left).

reduces annotation cost and increases quality, their approach is limited to crowdsourcing and moreover, still places a substantial burden on the study conductor.

Annotation curricula. Instead, we draw inspiration from traditional learning curricula (Kelly, 1977) and propose to apply them to annotation scenarios by considering annotators as learners. Our goal is to better guide annotators through the annotation task by ordering instances according to their difficulty, ensuring that simple instances are displayed first. As annotators gradually familiarize themselves with the task, we move on to more difficult instances. A major advantage such annotation curricula is that they are scalable to single or multi-batch scenarios and moreover, can be adapted to suit the annotator's needs (experts and crowd workers). We first formalize annotation curricula as a flexible framework and then conduct experiments using existing datasets and a carefully designed user study in which we investigate the following hypothesis:

Annotators that are presented with easy instances first and then with instances that gradually increase in terms of annotation difficulty require less annotation time or have improved annotation quality compared to annotators that receive the same instances in a random order (cf. Figure 5.2).

Our findings show that annotation order actually matters and that utilizing annotation curricula can significantly reduce annotation time.

5.4 Contributions

We explore possibilities of transferring our previous findings from Chapter 4 to scenarios beyond CALL. In Chapter 10, we showcase how interactive approaches can be implemented in real-world use cases by implementing a system that interactively collects an annotates dialogue data. We further point out important steps in the development of such systems concerning data privacy and safety regulations such as the GDPR (EU, 2016). Using these steps, we received full clearance by the ethics committee and staff council of TU Darmstadt. Finally, we contribute a German dataset consisting of 202 dialogues, 591 turns, and 1,027 annotated sentences collected in a highly specific domain (manufacturing at an assembly line in a factory).

In Chapter 11, we present annotation curricula as a novel paradigm for data annotation with several contributions. First, we provide a means to increase annotation efficiency with a low computational overhead, when using heuristic methods such as the BERTScore (Zhang et al., 2020). Second, we do not observe any drawbacks caused by ordering instances according to their annotation difficulty; all strategies outperform a random ordering. Finally, our work has raised increasing awareness among the community to consider instance ordering (Scholman et al., 2022; Eckman et al., 2024).

Part II

Publications

Chapter 6

Manipulating the Difficulty of C-Tests

Manipulating the Difficulty of C-Tests

Ji-Ung Lee and **Erik Schwan** and **Christian M. Meyer**
Ubiquitous Knowledge Processing (UKP) Lab and Research Training Group AIPHES
Computer Science Department, Technische Universität Darmstadt, Germany
https://www.ukp.tu-darmstadt.de

Abstract

We propose two novel manipulation strategies for increasing and decreasing the difficulty of C-tests automatically. This is a crucial step towards generating learner-adaptive exercises for self-directed language learning and preparing language assessment tests. To reach the desired difficulty level, we manipulate the size and the distribution of gaps based on absolute and relative gap difficulty predictions. We evaluate our approach in corpus-based experiments and in a user study with 60 participants. We find that both strategies are able to generate C-tests with the desired difficulty level.

1 Introduction

Learning languages is of utmost importance in an international society and formulated as a major political goal by institutions such as the European Council, who called for action to "teaching at least two foreign languages" (EC, 2002, p. 20). But also beyond Europe, there is a huge demand for language learning worldwide due to increasing globalization, digital communication, and migration.

Among multiple different learning activities required for effective language learning, we study one particular type of exercise in this paper: *C-tests* are a special type of cloze test in which the second half of every second word in a given text is replaced by a gap (Klein-Braley and Raatz, 1982). Figure 1 (a) shows an example. To provide context, the first and last sentences of the text do not contain any gaps. C-tests rely on the reduced redundancy principle (Spolsky, 1969) arguing that a language typically employs more linguistic information than theoretically necessary to communicate unambiguously. Proficient speakers intuitively understand an utterance even if the level of redundancy is reduced (e.g., when replacing a word's suffix with a gap), whereas learners typically rely on the redundant signal to extrapolate the meaning of an utterance.

Besides general vocabulary knowledge, C-tests require orthographic, morphologic, syntactic, and semantic competencies (Chapelle, 1994) to correctly fill in all gaps, which make them a frequently used tool for language assessment (e.g., placement tests). Given that C-tests can be easily generated automatically by introducing gaps into an arbitrary text and that there is usually only a single correct answer per gap given its context, C-tests are also relevant for self-directed language learning and massive open online courses (MOOC), where large-scale personalized exercise generation is necessary.

A crucial question for such tasks is predicting and manipulating the *difficulty* of a C-test. For language assessment, it is important to generate C-tests with a certain target difficulty to allow for comparison across multiple assessments. For self-directed language learning and MOOCs, it is important to adapt the difficulty to the learner's current skill level, as an exercise should be neither too easy nor too hard so as to maximize the learning effect and avoid boredom and frustration (Vygotsky, 1978). Automatic difficulty prediction of C-tests is hard, even for humans, which is why there have been many attempts to theoretically explain C-test difficulty (e.g., Sigott, 1995) and to model features used in machine learning systems for automatic difficulty prediction (e.g., Beinborn et al., 2014).

While state-of-the-art systems produce good prediction results compared to humans (Beinborn, 2016), there is yet no work on *automatically manipulating* the difficulty of C-tests. Instead, C-tests are generated according to a fixed scheme and manually post-edited by teachers, who might use the predictions as guidance. But this procedure is extremely time-consuming for language assessment and no option for large-scale self-directed learning.

In this paper, we propose and evaluate two strategies for automatically changing the gaps of a C-test in order to reach a given target difficulty. Our first

It i_ being fou___, moreover, i_ fairly cl___ correspondence wi__ the predi______ of t__ soothsayers o_ the th___ factories. Th__ predicted escal_____, and escal_____ is wh__ we a__ getting. T__ biggest nuc____ device t__ United Sta___ has expl____ measured so__ 15 meg. . .	It is being fought, more____, in fai___ cl___ corresp_______ with the predi______ of the sooth______ of the th___ fact_____. Th__ pred_____ escal_____, and escal_____ is what w_ are get____. The big____ nuc____ dev___ the United States h__ expl____ meas____ some 15 meg. . .	It i_ being fough_, moreover, i_ fairly clos_ correspondence wit_ the prediction_ of t__ soothsayers o_ the thin_ factories. The_ predicted escalatio_, and escalatio_ is wha_ we ar_ getting. T__ biggest nuclea_ device t__ United State_ has explode_ measured som_ 15 meg. . .
(a)	(b)	(c)

Figure 1: C-tests with (a) standard gap scheme, (b) manipulated gap position, and (c) manipulated gap size

strategy varies the distribution of the gaps in the underlying text and our second strategy learns to decide to increase or decrease a gap in order to make the test easier or more difficult. Our approach breaks away from the previously fixed C-test creation scheme and explores new ways of motivating learners by using texts they are interested in and generating tests from them at the appropriate level of difficulty. We evaluate our strategies both automatically and in a user study with 60 participants.

2 Related Work

In language learning research, there is vast literature on cloze tests. For example, Taylor (1953) studies the relation of cloze tests and readability. In contrast to C-tests (Klein-Braley and Raatz, 1982), cloze tests remove whole words to produce a gap leading to more ambiguous solutions.

Chapelle and Abraham (1990) contrast four types of cloze tests, including fixed-ratio cloze tests replacing every i^{th} word with a gap, rational cloze tests that allow selecting the words to replace according to the language trait that should be assessed, multiple-choice tests, and C-tests. Similar to our work, they conduct a user study and measure the difficulty posed by the four test types. They find that cloze tests replacing entire words with a gap are more difficult than C-tests or multiple-choice tests. In our work, we go beyond this by not only varying between gaps spanning the entire word (cloze test) or half of the word (C-test), but also changing the size of the C-test gaps. Laufer and Nation (1999) propose using C-tests to assess vocabulary knowledge. To this end, they manually construct C-tests with only a single gap, but use larger gaps than half of the word's letters. Our work is different to these previous works, since we test varying positions and sizes for C-test gaps and, more importantly, we aim at manipulating the difficulty of a C-test automatically by learning to predict the difficulty of the gaps and how their manipulation affects the difficulty.

Previous work on automatically controlling and manipulating test difficulty has largely focused on multiple-choice tests by generating appropriate distractors (i.e., incorrect solutions). Wojatzki et al. (2016) avoid ambiguity of their generated distractors, Hill and Simha (2016) fit them to the context, and Perez and Cuadros (2017) consider multiple languages. Further work by Zesch and Melamud (2014), Beinborn (2016), and Lee and Luo (2016) employ word difficulty, lexical substitution, and the learner's answer history to control distractor difficulty.

For C-tests, Kamimoto (1993) and Sigott (2006) study features of hand-crafted tests that influence the difficulty, and Beinborn et al. (2014) and Beinborn (2016) propose an automatic approach to estimate C-test difficulty, which we use as a starting point for our work.

Another related field of research in computer-assisted language learning is readability assessment and, subsequently, text simplification. There exists ample research on predicting the reading difficulty for various learner groups (Hancke et al., 2012; Collins-Thompson, 2014; Pilán et al., 2014). A specific line of research focuses on reducing the reading difficulty by text simplification (Chandrasekar et al., 1996). By reducing complex texts or sentences to simpler ones, more texts are made accessible for less proficient learners. This is done either on a word level by substituting difficult words with easier ones (e.g., Kilgarriff et al., 2014) or on a sentence level (Vajjala and Meurers, 2014). More recent work also explores sequence-to-sequence neural network architectures for this task (Nisioi et al., 2017). Although the reading difficulty of a text partly contributes to the overall exercise difficulty of C-tests, there are many other factors with a substantial influence (Sigott, 1995). In particular, we can generate many different C-tests from the same text and thus reading difficulty and text simplification alone are not sufficient to determine and manipulate the difficulty of C-tests.

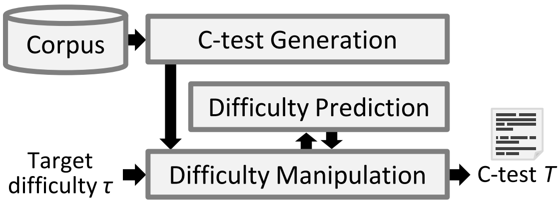


Figure 2: Proposed system architecture

3 Task Overview

We define a C-test $T = (u, w_1, \ldots, w_{2n}, v, G)$ as a tuple of left and right context u and v (typically one sentence) enframing $2n$ words w_i where $n = |G|$ is the number of gaps in the gap set G. In each gap $g = (i, \ell) \in G$, the last ℓ characters of word w_i are replaced by a blank for the learners to fill in. Klein-Braley and Raatz (1982) propose the default gap generation scheme DEF with $G = \{(2j, \lceil \frac{|w_{2j}|}{2} \rceil) \mid 1 \leq j \leq n\}$ in order to trim the (larger) second half of every second word. Single-letter words, numerals, and punctuation are not counted as words w_i and thus never contain gaps. Figure 1 (a) shows an example C-test generated with the DEF scheme.

A major limitation of DEF is that the difficulty of a C-test is solely determined by the input text. Most texts, however, yield a medium difficulty (cf. section 6) and thus do not allow any adaptation to beginners or advanced learners unless they are manually postprocessed. In this paper, we therefore propose two strategies to manipulate the gap set G in order to achieve a given *target difficulty* $\tau \in [0, 1]$ ranging from small values for beginners to high values for advanced learners. To estimate the difficulty $d(T) = \frac{1}{|G|} \sum_{g \in G} d(g)$ of a C-test T, we aggregate the predicted difficulty scores $d(g)$ of each gap. In section 4, we reproduce the system by Beinborn (2016) modeling $d(g) \approx e(g)$ as the estimated mean error rates $e(g)$ per gap across multiple learners, and we conduct additional validation experiments on a newly acquired dataset.

The core of our work is the manipulation of the gap set G in order to minimize the difference $|d(T) - \tau|$ between the predicted test difficulty $d(T)$ and the requested target difficulty τ. To this end, we employ our difficulty prediction system for validation and propose a new regression setup that predicts the relative change of $d(g)$ when manipulating the size ℓ of a gap.

Figure 2 shows our system architecture: Based on a text corpus, we generate C-tests for arbitrary texts (e.g., according to the learner's interests). Then, we manipulate the difficulty of the generated text by employing the difficulty prediction system in order to reach the given target difficulty τ for a learner (i.e., the estimated learner proficiency) to provide neither too easy nor too hard tests.

4 C-Test Difficulty Prediction

Beinborn et al. (2014) and Beinborn (2016) report state-of-the-art results for the C-test difficulty prediction task. However, there is yet no open-source implementation of their code and there is little knowledge about the performance of newer approaches. Therefore, we (1) conduct a reproduction study of Beinborn's (2016) system, (2) evaluate newer neural network architectures, and (3) validate the results on a newly acquired dataset.

Reproduction study. We obtain the original software and data from Beinborn (2016). This system predicts the difficulty $d(g)$ for each gap within a C-test using a support vector machine (SVM; Vapnik, 1998) with 59 hand-crafted features. The proposed features are motivated by four factors which are deemed important for assessing the gap difficulty: *item dependency*, *candidate ambiguity*, *word difficulty*, and *text difficulty*. We use the same data (819 filled C-tests), metrics, and setup as Beinborn (2016). That is, we perform leave-one-out cross validation (LOOCV) and measure the Pearson correlation ρ, the rooted mean squared error RMSE, and the quadratic weighted kappa qwκ as reported in the original work.

The left hand side of table 1 shows the results of our reproduced SVM compared to the original SVM results reported by Beinborn (2016). Even though we reuse the same code as in their original work, we observe small differences between our reproduction and the previously reported scores.

We were able to trace these differences back to libraries and resources which have been updated and thus changed over time. One example is Ubuntu's system dictionary, the *American English dictionary words* (wamerican), on which the original system relies. We experiment with different versions of the dictionary between Ubuntu 14.04 (wamerican v.7.1.1) and 18.04 (wamerican v.2018.04.16-1) and observe differences of one or two percentage points. As a best practice, we suggest to fix the versions of all resources and avoid any system dependencies.

Neural architectures. We compare the system with deep learning methods based on multi-layer

Model	Original data			New data		
	ρ	RMSE	qwκ	ρ	RMSE	qwκ
SVM (original)	.50	.23	.44	–	–	–
SVM (reproduced)	.49	.24	.47	.50	.21	.39
MLP	.42	.25	.31	.41	.22	.25
BiLSTM	.49	.24	.35	.39	.24	.27

Table 1: Results of the difficulty prediction approaches. SVM (original) has been taken from Beinborn (2016)

perceptrons (MLP) and bi-directional long short-term memory (BiLSTM) architectures, which are able to capture non-linear feature dependencies.[1] To cope for the non-deterministic behavior of the neural networks, we repeat all experiments ten times with different random weight initializations and report the averaged results (Reimers and Gurevych, 2017). While the MLP is trained similar as our reproduced SVM, the BiLSTM receives all gaps of a C-test as sequential input. We hypothesize that this sequence regression setup is better suited to capture gaps interdependencies. As can be seen from the table, the results of the neural architectures are, however, consistently worse than the SVM results. We analyze the RMSE on the train and development sets and observe a low bias, but a high variance. Thus, we conclude that although neural architectures are able to perform well for this task, they lack a sufficient amount of data to generalize.

Experiments on new data. To validate the results and assess the robustness of the difficulty prediction system, we have acquired a new C-test dataset from our university's language center. 803 participants of placement tests for English courses solved five C-tests (from a pool of 53 different C-tests) with 20 gaps each. Similar to the data used by Beinborn (2016), we use the error rates $e(g)$ for each gap as the $d(g)$ the methods should predict.

The right-hand side of table 1 shows the performance of our SVM and the two neural methods. The results indicate that the SVM setup is well-suited for the difficulty prediction task and that it successfully generalizes to new data.

Final model. We train our final SVM model on all available data (i.e., the original and the new data) and publish our source code and the trained model on GitHub.[2] Similar to Beinborn (2016), we cannot openly publish our dataset due to copyright.

Algorithm 1 Gap selection strategy (SEL)

procedure GAPSELECTION(T, τ)
 $G_{\text{FULL}} \leftarrow \{(i, \lceil \frac{|w_i|}{2} \rceil) \mid 1 \leq i \leq 2n\}$
 $G_{\text{SEL}} \leftarrow \emptyset$
 while $|G_{\text{SEL}}| < n$ **do**
  $G_{\leq\tau} \leftarrow \{g \in G_{\text{FULL}} \mid d(g) \leq \tau\}$
  if $|G_{\leq\tau}| > 0$ **then**
   $g^* \leftarrow \arg\min_{g \in G_{\leq\tau}} |d(g) - \tau|$
   $G_{\text{SEL}} \leftarrow G_{\text{SEL}} \cup \{g^*\}$
   $G_{\text{FULL}} \leftarrow G_{\text{FULL}} \setminus \{g^*\}$
  $G_{>\tau} \leftarrow \{g \in G_{\text{FULL}} \mid d(g) > \tau\}$
  if $|G_{>\tau}| > 0$ **then**
   $g^* \leftarrow \arg\min_{g \in G_{>\tau}} |d(g) - \tau|$
   $G_{\text{SEL}} \leftarrow G_{\text{SEL}} \cup \{g^*\}$
   $G_{\text{FULL}} \leftarrow G_{\text{FULL}} \setminus \{g^*\}$
 return G_{SEL}

5 C-Test Difficulty Manipulation

Given a C-test $T = (u, w_1, \ldots, w_{2n}, v, G)$ and a target difficulty τ, the goal of our manipulation strategies is to find a gap set G such that $d(T)$ approximates τ. A naïve way to achieve this goal would be to generate C-tests for all texts in a large corpus with the DEF scheme and use the one with minimal $|d(T) - \tau|$. However, most corpora tend to yield texts of a limited difficulty range that only suit a specific learner profile (cf. section 6). Another drawback of the naïve strategy is that it is difficult to control for the topic of the underlying text and in the worst case, the necessity to search through a whole corpus for selecting a fitting C-test.

In contrast to the naïve strategy, our proposed manipulation strategies are designed to be used in real time and manipulate any given C-test within 15 seconds at an acceptable quality.[3] Both strategies operate on a given text (e.g., on a topic a learner is interested in) and manipulate its gap set G in order to come close to the learner's current language skill. The first strategy varies the position of the gaps and the second strategy learns to increase or decrease the size of the gaps.

5.1 Gap Selection Strategy

The default C-test generation scheme DEF creates a gap in every second word w_{2j}, $1 \leq j \leq n$. The core idea of our first manipulation strategy SEL is to distribute the n gaps differently among the all $2n$ words in order to create gaps for easier or harder words than in the default generation scheme. Therefore, we use the difficulty predic-

[1]Network parameters and a description of the tuning process are provided in this paper's appendix.

[2]https://github.com/UKPLab/acl2019-ctest-difficulty-manipulation (licensed under the Apache License 2.0).

[3]On an *Intel-i5* with 4 CPUs and 16 GB RAM.

tion system to predict $d(g)$ for any possible gap $g \in G_{\texttt{FULL}} = \{(i, \lceil\frac{|w_i|}{2}\rceil) \mid 1 \leq i \leq 2n\}$ (i.e., assuming a gap in all words rather than in every second word). Then, we alternate between adding gaps to the resulting $G_{\texttt{SEL}}$ that are easier and harder than the preferred target difficulty τ, starting with those having a minimal difference $|d(g) - \tau|$.

Algorithm 1 shows this procedure in pseudocode and figure 1 shows a C-test whose difficulty has been increased with this strategy. Note that it has selected gaps at *corresponding* rather than *with*, and *soothsayers* rather than *the*. Our proposed algorithm is optimized for runtime. An exhaustive search would require testing $\binom{2n}{n}$ combinations if the number of gaps is constant. For $n = 20$, this yields 137 billion combinations. While more advanced optimization methods might find better gap selections, we show in section 6 that our strategy achieves good results.

5.2 Gap Size Strategy

Our second manipulation strategy SIZE changes the size of the gaps based on a pre-defined gap set. Increasing a gap $g = (i, \ell)$ by one or more characters, yielding $g' = (i, \ell + k)$ increases its difficulty (i.e., $d(g') \geq d(g)$), while smaller gaps make the gap easier. We identify a major challenge in estimating the effect of increasing or decreasing the gap size on the gap difficulty. Although $d(g')$ could be estimated using the full difficulty prediction system, the search space is even larger than for the gap selection strategy, since each of the n gaps has $|w_i| - 2$ possible gap sizes to test. For $n = 20$ and an average word length of six, this amounts to one trillion possible combinations.

We therefore propose a new approach to predict the *relative difficulty change* of a gap $g = (i, \ell)$ when increasing the gap size by one letter $\Delta_{\text{inc}}(g) \approx d(g') - d(g)$, $g' = (i, \ell + 1)$ and correspondingly when decreasing the gap size by one letter $\Delta_{\text{dec}}(g) \approx d(g) - d(g')$, $g' = (i, \ell - 1)$. The notion of relative difficulty change enables gap size manipulation in real time, since we do not have to invoke the full difficulty prediction system for all combinations. Instead, we can incrementally predict the effect of changing a single gap.

To predict Δ_{inc} and Δ_{dec}, we train two SVMs on all gap size combinations of 120 random texts from the Brown corpus (Francis, 1965) using the following features: predicted absolute gap difficulty, word length, new gap size, modified character, a binary indicator if the gap is at a *th* sound, and logarithmic difference of alternative solutions capturing the degree of ambiguity with varying gap size.

Algorithm 2 Gap size strategy (SIZE)

procedure INCREASEDIFFICULTY(T, τ)
  $G_{\texttt{SIZE}} \leftarrow G_{\texttt{DEF}}$
  $D \leftarrow d(T)$
  while $D < \tau$ **do**
    $g^* = (i, \ell) \leftarrow \arg\max_{g \in G_{\texttt{SIZE}}} \Delta_{\text{inc}}(g)$
    $\ell \leftarrow \ell + 1$
    $D \leftarrow D + \Delta_{\text{inc}}(g)$
  return $G_{\texttt{SIZE}}$

With a final set of only six features, our new models are able to approximate the relative difficulty change very well deviating from the original system's prediction only by 0.06 RMSE for Δ_{inc} and 0.13 RMSE for Δ_{dec}. The predictions of both models highly correlate with the predictions achieving a Pearson's ρ of over 0.8. Besides achieving a much faster average runtime of 0.056 seconds for the relative model vs. 11 seconds for the full prediction of a single change, we can invoke the relative model iteratively to estimate $d(T)$ for multiple changes of the gap size more efficiently.

The final manipulation strategy then requires just a single call of the full prediction system. If $d(T) < \tau$, we incrementally increase the gap sizes to make T more difficult and, vice-versa, decrease the gap sizes if $d(T) > \tau$. In each iteration, we modify the gap with the highest relative difficulty change in order to approach the given target difficulty τ as quickly as possible. Algorithm 2 shows pseudocode for creating G_{size} with increased difficulty (i.e., $d(T) < \tau$) based on the default gap scheme DEF. The procedure for $d(T) > \tau$ works analogously, but using Δ_{dec} and decreasing the gap size. Figure 1 (c) shows a much easier version of the example C-test, in which a learner often only has to complete the last one or two letters.

6 Evaluation of the Manipulation System

To evaluate our C-test manipulation strategies, we first test their ability to cover a higher range of target difficulties than the default generation scheme and then measure how well they meet the desired target difficulty for texts from different domains. We conduct our experiments on 1,000 randomly chosen paragraphs for each of the Gutenberg (Lahiri, 2014), Reuters (Lewis et al., 2004), and Brown (Francis, 1965) corpora. We conduct our experiments on English, but our strategies can be adapted to many related languages.

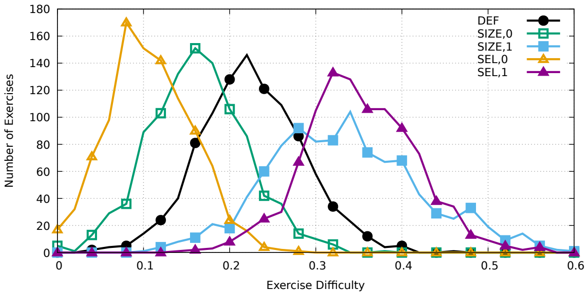


Figure 3: Difficulty distribution of exercises generated with `DEF`, `SEL`, and `SIZE` for extreme τ values

Difficulty range. The black ●-marked line of figure 3 shows the distribution of $d(T)$ based on our difficulty prediction system when creating a C-test with the default generation scheme `DEF` for all our samples of the Brown corpus. The vast majority of C-tests range between 0.15 and 0.30 with a predominant peak at 0.22.

To assess the maximal difficulty range our strategies can achieve, we generate C-tests with maximal ($\tau = 1$) and minimal target difficulty ($\tau = 0$) for both strategies $S \in \{\texttt{SEL}, \texttt{SIZE}\}$, which are also shown in figure 3 as (S, τ). Both strategies are able to clearly increase and decrease the test difficulty in the correct direction and they succeed in substantially increasing the total difficulty range beyond `DEF`. While `SEL` is able to reach lower difficulty ranges, it has bigger issues with generating very difficult tests. This is due to its limitation to the fixed gap sizes, whereas `SIZE` can in some cases create large gaps that are ambiguous or even unsolvable. Since `SIZE` is, however, limited to the 20 predefined gaps, it shows a higher variance. Especially short gaps such as *is* and *it* cannot be made more difficult. Combining the two strategies is thus a logical next step for future work, building upon our findings for both strategies. We make similar observations on the Reuters and Gutenberg corpora and provide the respective figures in the appendix.

Manipulation quality. We finally evaluate how well each strategy S reaches a given target difficulty. That is, we sample a random corpus text and τ, create the C-test using strategy S, predict the test difficulty $d(T)$ and measure its difference to τ using RMSE. Table 2 shows the results for our three corpora. Throughout all three corpora, both manipulation strategies perform well. `SEL` consistently outperforms `SIZE`, which matches our observations from the previous experiment. Mind that these results depend on the quality of the automatic difficulty predictions, which is why we conduct a user-based evaluation in the next section.

Strategy	Brown	Reuters	Gutenberg
`SEL`	.11	.12	.10
`SIZE`	.13	.15	.12

Table 2: RMSE for both strategies on each corpora with randomly sampled target difficulties τ

7 User-based Evaluation

Hypothesis. To evaluate the effectiveness of our manipulation strategies in a real setting, we conduct a user study and analyze the difficulty of the manipulated and unmanipulated C-tests. We investigate the following hypothesis: When increasing a test's difficulty using strategy S, the participants will make more errors and judge the test harder than a default C-test and, vice versa, when decreasing a test's difficulty using S, the participants will make less errors and judge the test easier.

Experimental design. We select four different English texts from the Brown corpus and shorten them to about 100 words with keeping their paragraph structure intact. None of the four texts is particularly easy to read with an average grade level above 12 and a Flesh reading ease score ranging between 25 (very difficult) to 56 (fairly difficult). In the supplementary material, we provide results of an automated readability analysis using standard metrics. From the four texts, we then generate the C-tests T_i, $1 \leq i \leq 4$ using the default generation scheme `DEF`. All tests contain exactly $n = 20$ gaps and their predicted difficulties $d(T_i)$ are in a mid range between 0.24 and 0.28. T_1 remains unchanged in all test conditions and is used to allow the participants to familiarize with the task. For the remaining three texts, we generate an easier variant $T_i^{S,\text{dec}}$ with target difficulty $\tau = 0.1$ and a harder variant $T_i^{S,\text{inc}}$ with $\tau = 0.5$ for both strategies $S \in \{\texttt{SEL}, \texttt{SIZE}\}$.

From these tests, we create 12 sequences of four C-tests that we give to the participants. Each participant receives T_1 first to familiarize with the task. Then, they receive one easy $T_i^{S,\text{dec}}$, one default T_i, and one hard $T_i^{S,\text{inc}}$ C-test for the same strategy S based on the texts $i \in \{2, 3, 4\}$ in random order without duplicates (e.g., the sequence $T_1\, T_2^{\texttt{SEL,dec}}\, T_3\, T_4^{\texttt{SEL,inc}}$). Having finished a C-test, we ask them to judge the difficulty of this test on a five-point Likert scale ranging from *too easy* to *too hard*. After solving the last test, we additionally collect a ranking of all four tests by their difficulty.

Data collection. We collect the data from our participants with a self-implemented web interface for solving C-tests. We create randomized credentials linked to a unique ID for each participant and obfuscate their order, such that we can distinguish them but cannot trace back their identity and thus avoid collecting any personal information. Additionally, we ask each participant for their consent on publishing the collected data. For experiments with a similar setup and task, we obtained the approval of the university's ethics commission. After login, the participants receive instructions and provide a self-assessment of their English proficiency and their time spent on language learning. The participants then solve the four successive C-tests without knowing the test difficulty or the manipulation strategy applied. They are instructed to spend a maximum of five minutes per C-test to avoid time-based effects and to prevent them from consulting external resources, which would bias the results.

Participants. A total of 60 participants completed the study. We uniformly distributed the 12 test sequences (six per strategy), such that we have 30 easy, 30 default, and 30 hard C-test results for each manipulation strategy. No participant is native in English, 17 are taking language courses, and 57 have higher education or are currently university students. The frequency of their use of English varies, as we found a similar number of participants using English daily, weekly, monthly, and (almost) never in practice. An analysis of the questionnaire is provided in the paper's appendix.

Hypothesis testing. We evaluate our hypothesis along three dimensions: (1) the actual error rate of the participants, (2) the perceived difficulty after each individual C-test (Likert feedback), and (3) the participants' final difficulty ranking. While the latter forces the participants to provide an explicit ranking, the former allows them to rate C-tests equally difficult. We conduct significance testing at the Bonferroni-corrected $\alpha = \frac{0.05}{2} = 0.025$ for each dimension using one-tailed t-tests for the continuous error rates and one-tailed Mann–Whitney U tests for the ordinal-scaled perceived difficulties and rankings. Figure 4 shows notched boxplots of our results.

To test our hypothesis, we first formulate a null hypothesis that (a) the mean error rate, (b) the median perceived difficulty (Likert feedback), and (c) the median rank of the manipulated tests equal the default tests. While the participants have an average error rate of 0.3 on default C-tests, the $T_i^{S,\mathrm{dec}}$ tests are significantly easier with an average error rate of 0.15 ($t = 7.49$, $p < 10^{-5}$) and the $T_i^{S,\mathrm{inc}}$ tests are significantly harder with an average error rate of 0.49 ($t = -7.83$, $p < 10^{-5}$), so we can safely reject the null hypothesis for error rates.

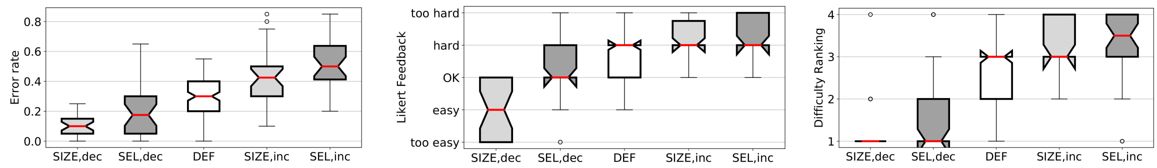
	easy (dec)		default	hard (inc)	
	`SEL`	`SIZE`	`DEF`	`SEL`	`SIZE`
T_1	–	–	.30	–	–
T_2	.17*	.11*	.34	.66*	.44*
T_3	.16*	.10*	.27	.52*	.43*
T_4	.28	.09*	.30	.43*	.45*
Average	.20*	.10*	.30	.53*	.44*

Table 3: Mean error rates $e(T)$ per text and strategy. Results marked with * deviate significantly from `DEF`

Table 3 shows the error rates per C-test and strategy. Both `SEL` and `SIZE` are overall able to significantly ($p < 0.025$) increase and decrease the test's difficulty over `DEF`, and with the exception of $T_4^{\texttt{SEL},\mathrm{dec}}$, the effect is also statistically significant for all individual text and strategy pairs. Figure 5 shows the 30 participants per strategy on the x-axis and their error rates in their second to fourth C-test on the y-axis. C-tests, for which we increased the difficulty (S, inc), yield more errors than C-tests with decreased difficulty (S, dec) in all cases. The easier tests also yield less errors than the test with the default scheme `DEF` in most cases. While hard tests often have a much higher error rate than `DEF`, we find some exceptions, in which the participant's error rate is close or even below the `DEF` error rate.

Regarding the perceived difficulty, we find that the participants judge the manipulated C-tests with lower $d(T)$ as easier on both the Likert scale ($z = 6.16$, $p < 10^{-5}$) and in the rankings ($z = 6.59$, $p < 10^{-5}$) based on the Mann-Whitney-U test. The same is true for C-tests that have been manipulated to a higher difficulty level, which the participant judge harder ($z = -4.57$, $p < 10^{-5}$) and rank higher ($z = -3.86$, $p < 6 \cdot 10^{-5}$). We therefore reject the null hypotheses for the Likert feedback and the rankings and conclude that both strategies can effectively manipulate a C-test's difficulty.

Manipulation quality. We further investigate if the strategies yield different difficulty levels. There-



(a) (b) (c)

Figure 4: Notched boxplots for the (a) observed error rates, (b) Likert feedback, and (c) the participants' rankings

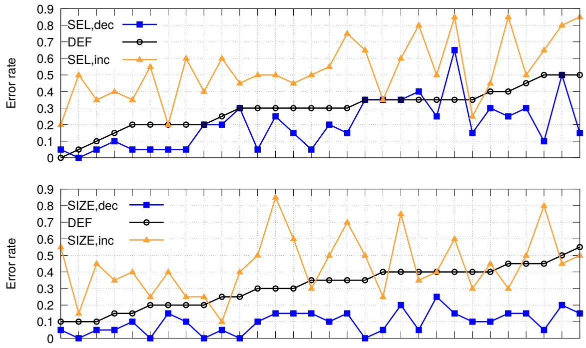


Figure 5: Error rates per participant and strategy

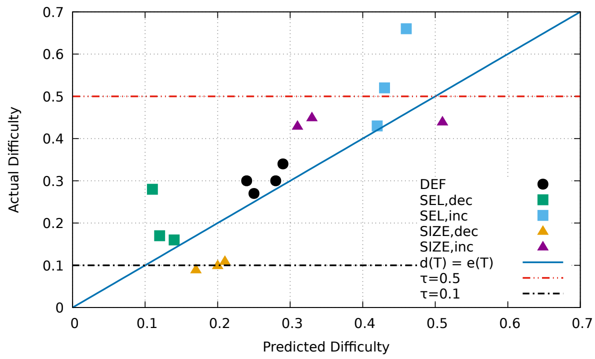


Figure 6: Predicted difficulties $d(T)$ vs the actual error rates $e(T)$.

	SEL		DEF	SIZE	
τ	.10	.50	–	.10	.50
RMSE(e, d)	.10	.13	.04	.09	.11
RMSE(e, τ)	.12	.10	–	.01	.06

Table 4: RMSE between the actual difficulty $e(T)$ and predicted difficulty $d(T)$ as well as target difficulty τ.

fore, we use two-tailed significance testing between SEL and SIZE for all three dimensions. We find that SIZE yields significantly easier C-tests than SEL in terms of error rates ($p = 0.0014$) and Likert feedback ($p = 6 \cdot 10^{-5}$), and observe $p = 0.0394$ for the rankings. For increasing the difficulty, we, however, do not find significant differences between the two strategies. Since both strategies successfully modify the difficulty individually, this motivates research on combined strategies in the future.

We furthermore investigate how well our strategies perform in creating C-tests with the given target difficulty τ. Table 4 shows the RMSE for $e(T)$ and $d(T)$ as well as for $e(T)$ and τ for both strategies. As expected, our difficulty prediction system works best for C-tests generated with DEF as they use the same scheme as C-tests in the training data. Though slightly worse than for DEF, we still find very low RMSE scores for manipulated C-tests. This is especially good when considering that the system's performance on our newly acquired dataset yields and RMSE of 0.21 (cf. section 6). Computing the RMSE with respect to our chosen target difficulties τ yields equally good results for SEL and exceptionally good results for SIZE. Figure 6 displays $d(T)$ in comparison to $e(T)$ for each individual text and strategy. With the exception of $T_2^{\text{SEL,inc}}$ and $T_4^{\text{SEL,dec}}$, all predictions are close to the optimum (i.e., the diagonal) and also close to the desired target difficulty τ.

In a more detailed analysis, we find two main sources of problems demanding further investigation: First, the difficulty prediction quality when deviating from DEF and second, the increasing ambiguity in harder C-tests. However, it underestimates the $d(T) = 0.11$ for $T_4^{\text{SEL,dec}}$ (the same text used in figure 1), for which we found an actual error rate of 0.28. This is due to chains of four successive gaps, such as:

gap g	i_	wh__	w_	a__
solution	is	what	we	are
$d(g)$	0.17	0.22	0.23	0.19
$e(g)$	0.70	0.40	0.10	0.20

As the prediction system has been trained only on DEF-generated C-tests, it underestimates $d(g)$ for cases with limited context. It will be interesting for future work to focus on modeling gap interdependencies in C-tests deviating from DEF.

Another issue we observe is that the gap size strategy might increase the ambiguity of the C-test. In the standard scheme, there is in most cases only a single correct answer per gap. In $T_2^{\text{SIZE,inc}}$, however, the SIZE strategy increased the gap of the word *professional* to its maximal length yielding p___________. One participant answered *popularising* for this gap, which also fits the given context. We carefully checked our datasetfor other ambiguity, but only found one additional case: In T_4, instead of the word *close*, 13 participants out of 30 used *clear* as a modifier of *correspondence*, which both produce meaningful contexts. Given that this case is already ambiguous in the DEF scheme yielding the gap cl___, we conclude that the issue is not severe, but that the difficulty prediction system should be improved to better capture ambiguous cases; for example, by introducing collocational features weighted by their distribution within a corpus into Δ_{inc} and Δ_{dec}.

8 Conclusion

In this work, we proposed two novel strategies for automatically manipulating the difficulty of C-test exercises. Our first strategy selects which words should be turned into a gap, and the second strategy learns to increase or decrease the size of the gaps. Both strategies automatically predict the difficulty of a test to make informed decisions. To this end, we reproduced previous results, compared them to neural architectures, and tested them on a newly acquired dataset. We evaluate our difficulty manipulation pipeline in a corpus-based study and with real users. We show that both strategies can effectively manipulate the C-test difficulty, as both the participants' error rates and their perceived difficulty yield statistically significant effects. Both strategies reach close to the desired difficulty level.

Our error analysis points out important directions for future work on detecting ambiguous gaps and modeling gap interdependencies for C-tests deviating from the default generation scheme. An important observation is that manipulating the gaps' size and position does not only influence the C-test difficulty, but also addresses different competencies (e.g., requires more vocabulary knowledge or more grammatical knowledge). Future manipulation strategies that take the competencies into account have the potential to train particular skills and to better control the competencies required for a placement test. Another strand of research will be combining both strategies and deploying the manipulation strategies in a large scale testing platform that allows the system to adapt to an individual learner over time. A core advantage of our manipulation strategies is that we can work with any given text and thus provide C-tests that do not only have the desired difficulty, but also integrate the learner's interest or the current topic of a language course.

Acknowledgments

This work has been supported by the Hessian research excellence program "Landes-Offensive zur Entwicklung Wissenschaftlich-ökonomischer Exzellenz" (LOEWE) as part of the *a! – automated language instruction* project under grant No. 521/17-03 and by the German Research Foundation as part of the Research Training Group "Adaptive Preparation of Information from Heterogeneous Sources" (AIPHES) under grant No. GRK 1994/1. We thank the anonymous reviewers for their detailed and helpful comments. We furthermore thank the language center of the Technische Universität Darmstadt for their cooperation and Dr. Lisa Beinborn for providing us with the code for our reproduction study.

References

Lisa Beinborn, Torsten Zesch, and Iryna Gurevych. 2014. Predicting the Difficulty of Language Proficiency Tests. *Transactions of the Association for Computational Linguistics*, 2:517–529.

Lisa Marina Beinborn. 2016. *Predicting and manipulating the difficulty of text-completion exercises for language learning*. Ph.D. thesis, Technische Universität Darmstadt.

Raman Chandrasekar, Christine Doran, and Bangalore Srinivas. 1996. Motivations and methods for text simplification. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING): Volume 2*, pages 1041–1044, Copenhagen, Denmark.

C. A. Chapelle. 1994. Are C-tests valid measures for L2 vocabulary research? *Second Language Research*, 10(2):157–187.

Carol A. Chapelle and Roberta G. Abraham. 1990. Cloze method: what difference does it make? *Language Testing*, 7(2):121–146.

Kevyn Collins-Thompson. 2014. Computational assessment of text readability: A survey of current and future research. *International Journal of Applied Linguistics – Special Issue on Recent Advances in Automatic Readability Assessment and Text Simplification*, 165(2):97–135.

EC. 2002. Presidency Conclusions. Barcelona European Council 15 and 16 March 2002. Report SN 100/1/02 REV 1, Council of the European Union.

W. Nelson Francis. 1965. A standard corpus of edited present-day american english. *College English*, 26(4):267–273.

Julia Hancke, Sowmya Vajjala, and Detmar Meurers. 2012. Readability classification for german using lexical, syntactic, and morphological features. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING)*, pages 1063–1080, Mumbai, India.

Jennifer Hill and Rahul Simha. 2016. Automatic generation of context-based fill-in-the-blank exercises using co-occurrence likelihoods and google n-grams. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications (BEA)*, pages 23–30, San Diego, CA, USA.

Tadamitsu Kamimoto. 1993. Tailoring the Test to Fit the Students: Improvement of the C-Test through Classical Item Analysis. *Language Laboratory*, 30:47–61.

Adam Kilgarriff, Frieda Charalabopoulou, Maria Gavrilidou, Janne Bondi Johannessen, Saussan Khalil, Sofie Johansson Kokkinakis, Robert Lew, Serge Sharoff, Ravikiran Vadlapudi, and Elena Volodina. 2014. Corpus-based vocabulary lists for language learners for nine languages. *Language Resources and Evaluation*, 48(1):121–163.

Christine Klein-Braley and Ulrich Raatz. 1982. Der C-Test: ein neuer Ansatz zur Messung allgemeiner Sprachbeherrschung. *AKS-Rundbrief*, 4:23–37.

Shibamouli Lahiri. 2014. Complexity of Word Collocation Networks: A Preliminary Structural Analysis. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 96–105, Gothenburg, Sweden.

Batia Laufer and Paul Nation. 1999. A vocabulary-size test of controlled productive ability. *Language Testing*, 16(1):33–51.

John Lee and Mengqi Luo. 2016. Personalized exercises for preposition learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL): System Demonstrations*, pages 115–120, Berlin, Germany.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. RCV1: A New Benchmark Collection for Text Categorization Research. *Journal of Machine Learning Research*, 5(Apr):361–397.

Sergiu Nisioi, Sanja Štajner, Simone Paolo Ponzetto, and Liviu P. Dinu. 2017. Exploring neural text simplification models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL): Short Papers*, volume 2, pages 85–91, Vancouver, Canada.

Naiara Perez and Montse Cuadros. 2017. Multilingual call framework for automatic language exercise generation from free text. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL): Software Demonstrations*, pages 49–52, Valencia, Spain.

Ildikó Pilán, Elena Volodina, and Richard Johansson. 2014. Rule-based and machine learning approaches for second language sentence-level readability. In *Proceedings of the Ninth Workshop on Innovative Use of NLP for Building Educational Applications (BEA)*, pages 174–184, Baltimore, MD, USA.

Nils Reimers and Iryna Gurevych. 2017. Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 338–348, Copenhagen, Denmark.

Günther Sigott. 1995. The C-Test: Some Factors of Difficulty. *AAA: Arbeiten aus Anglistik und Amerikanistik*, 20(1):43–53.

Günther Sigott. 2006. How fluid is the c-test construct? In *Der C-Test: Theorie, Empirie, Anwendungen – The C-Test: Theory, Empirical Research, Applications*, Language Testing and Evaluation, pages 139–146. Frankfurt am Main: Peter Lang.

Bernard Spolsky. 1969. Reduced Redundancy as a Language Testing Tool. In G.E. Perren and J.L.M. Trim, editors, *Applications of linguistics*, pages 383–390. Cambridge: Cambridge University Press.

Wilson L. Taylor. 1953. "Cloze Procedure": A New Tool for Measuring Readability. *Journalism Bulletin*, 30(4):415–433.

Sowmya Vajjala and Detmar Meurers. 2014. Assessing the relative reading level of sentence pairs for text simplification. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 288–297, Gothenburg, Sweden.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. New York: Wiley.

Lev Vygotsky. 1978. *Mind in society: The development of higher psychological processes*. Cambridge: Harvard University Press.

Michael Wojatzki, Oren Melamud, and Torsten Zesch. 2016. Bundled gap filling: A new paradigm for unambiguous cloze exercises. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications (BEA)*, pages 172–181, San Diego, CA, USA.

Torsten Zesch and Oren Melamud. 2014. Automatic generation of challenging distractors using context-sensitive inference rules. In *Proceedings of the*

Ninth Workshop on Innovative Use of NLP for Building Educational Applications (BEA), pages 143–148, Baltimore, MD, USA.

Manipulating the Difficulty of C-Tests
– Supplementary Material –

Ji-Ung Lee and **Erik Schwan** and **Christian M. Meyer**
Ubiquitous Knowledge Processing (UKP) Lab and Research Training Group AIPHES
Computer Science Department, Technische Universität Darmstadt, Germany
https://www.ukp.tu-darmstadt.de

This document provides supplementary material for our ACL 2019 paper "Manipulating the Difficulty of C-Tests".

1 C-Test Difficulty Manipulation

Feature description for Δ_{inc} and Δ_{dec}. We provide an extended feature description for the subset of features used for our relative difficulty prediction models Δ_{inc} and Δ_{dec}. Features marked with * are also used by the absolute difficulty prediction model proposed by Beinborn (2016). For a gap $g = (i, \ell)$ in word w_i, we define:

- the predicted absolute gap difficulty $d(g)$ for the initial C-test created with DEF obtained from our reproduced difficulty prediction system, see line 3 of algorithm 2 (PS),
- the word length $|w_i|$ (WL*),
- the new gap size $\ell \pm 1$ after modification (GL*),
- the modified character $w_i[\ell]$ when increasing or decreasing the gap (CH),
- a binary indicator if the gap is after a *th* sound (RG*), and
- the logarithmic difference of alternative solutions (LD*) capturing the change in the degree of ambiguity when increasing or decreasing ℓ.

Feature ablation test. We conduct feature ablation tests to evaluate the impact of each feature on our relative difficulty prediction models Δ_{inc} and Δ_{dec}. Both models were evaluated on all gap size combinations for 120 random texts from the Brown corpus (Francis, 1965) with a three-fold cross-validation. Table 1 shows the performance increase for each model after including each feature. RMSE shows the deviation and ρ the correlation of our relative difficulty prediction compared to the absolute difficulty prediction. Although the increase in performance with RG is not substantial, we decided to include it as a meaningful feature which measures the impact for increasing or decreasing the gap size in words starting with *th*.

	Δ_{inc}		Δ_{dec}	
Feature	RMSE	ρ	RMSE	ρ
PS	.088	.521	.213	.271
+ WL	.072	.712	.183	.570
+ GL	.066	.771	.162	.687
+ CH	.069	.735	.157	.707
+ RG	.069	.736	.157	.707
+ LD	.061	.805	.131	.806

Table 1: Feature ablation test for Δ_{inc} and Δ_{dec} compared to the full difficulty prediction system

2 Neural Network Parameters

Although obtaining state-of-the-art results in many tasks, the deep neural networks we evaluated during our preliminary experiments did perform worse than the SVM. We performed parameter tuning with 100 randomly initialized configurations for both, MLP and BiLSTM. We tune the following parameters:

- Number of hidden layers $H_l \in [1, ..., 5]$
- Number of hidden units $H_l^u \in [50, ..., 200]$
- Dropout rate $D_x \in [0.1, ..., 0.5]$

We use Adam with Nesterov Momentum (Dozat, 2016) as our optimizer and keep the batch size at 5 for both models. All models are trained for 200 epochs with an early stopping after 10 epochs with no improvement of the loss. Figure 1 shows the resulting architectures of both models after tuning. Since our goal is to output regression values, we use a linear activation function in the output layer.

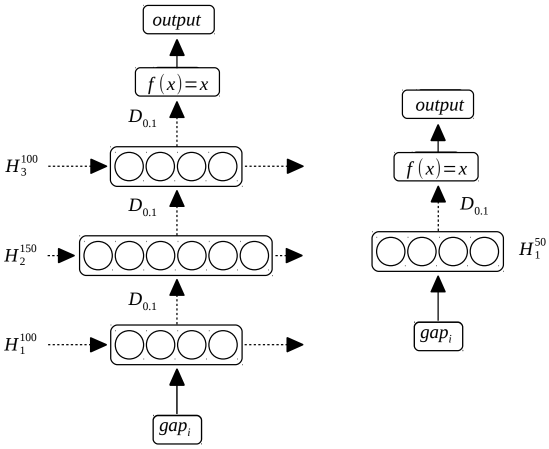


Figure 1: Final, tuned architectures of our BiLSTM (left) and MLP (right) models.

In preliminary experiments, we also tuned and evaluated BiLSTMs including soft attention, however, they performed even worse than the models without any attention. Analyzing the results of the best performing attention based model showed that it had a strong bias towards predicting the mean value of the whole training set. Furthermore, similar to the other neural models, it showed a low error on the training set (low bias) and a rather high error on the development set (high variance), indicating a lack of training data.

3 Evaluation of the Manipulation System

Results for additional corpora. Figure 2 and figure 3 show our results on the Gutenberg (Lahiri, 2014) and the Reuters (Lewis et al., 2004) corpora. As already discussed in the main paper, we observe very similar distributions for DEF, SEL, and SIZE across both corpora matching our descriptions for the Brown (Francis, 1965) corpus.

We further compute $\tau_{\max} - \tau_{\min}$ for SEL and SIZE for each text within a corpus and thus, measure the difficulty range both strategies are able to cover for a single text. As figure 4 shows, SEL achieves a larger difficulty range, whereas considerably more C-tests achieve higher difficulty levels when generated with SIZE. We again observe very similar distributions throughout the three corpora.

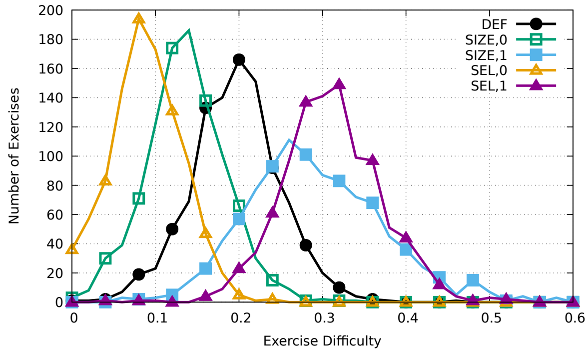


Figure 2: Difficulty distribution of exercises generated with DEF, SEL, and SIZE for extreme τ values on the Gutenberg corpus.

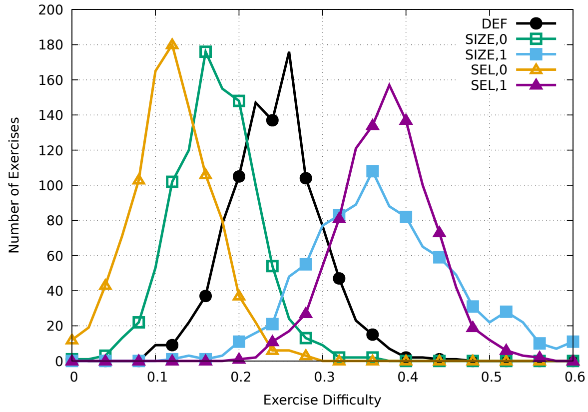


Figure 3: Difficulty distribution of exercises generated with DEF, SEL, and SIZE for extreme τ values on the Reuters corpus.

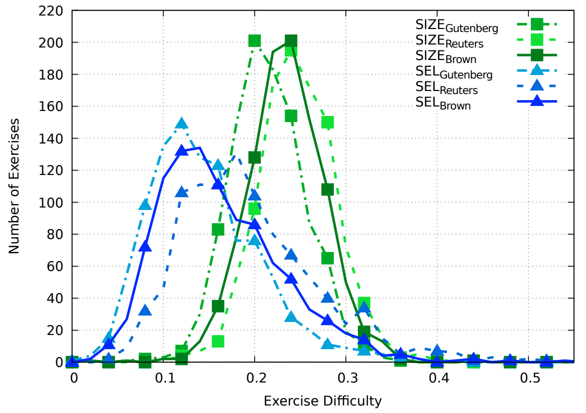


Figure 4: Error rate range ($\tau_{\max} - \tau_{\min}$) of exercises generated with SEL and SIZE for all three corpora.

4 User-based Evaluation

Questionnaire. At the begin of our study, our participants answered a questionnaire for a self-assessment of their English proficiency described in figure 5. We partitioned our questionnaire into three sections asking about 1) our participants' *English proficiency* (**Q1**, **Q2**), 2) their *learning habits and goals* (**Q4**), and 3) *other languages* they have been learning (**Q3**, **Q5**, **Q6**).

Q1: Please estimate your current language proficiency in English
A1: ○ *Beginner (A1)* ○ *Elementary (A2)* ○ *Intermediate (B1)* ○ *Upper Intermediate (B2)* ○ *Advanced (C1)* ○ *Proficiency (C2)*

Q2: I studied English for about ___ years.

Q3: Do you participate in any language learning courses (for example, at your university, evening school,...)? If yes, than which ones?
A3: ○ *Yes,* ________. ○ *No.*

Q4: How often do you practice English?
A4: ○ *Never* ○ *Monthly* ○ *Weekly* ○ *Daily*

Q5: What is your native language?
A5: ______________________

Q6: Have you tried learning other languages before? If yes, than which ones?
A6: ○ *Yes,* ________. ○ *No.*

Figure 5: Self-assessment questionnaire.

Answers. As described in the main paper, 17 participants are taking in language courses (**Q3**). Overall, 41 participants have tried to learn a second language (**Q6**). The exact answers can be found in the data we provide. Note, that not all participants provided the language which they attempted to learn since this was not mandatory. Figure 6–8 shows our participants' answers to **Q1**, **Q2**, and **Q4**. As can be seen, none of our participants consider themselves at the *Beginner (A1)* level. Furthermore, most of them are rather confident in their English proficiency and provide an estimate of either *Upper Intermediate (B2)* or *Advanced (C1)*.

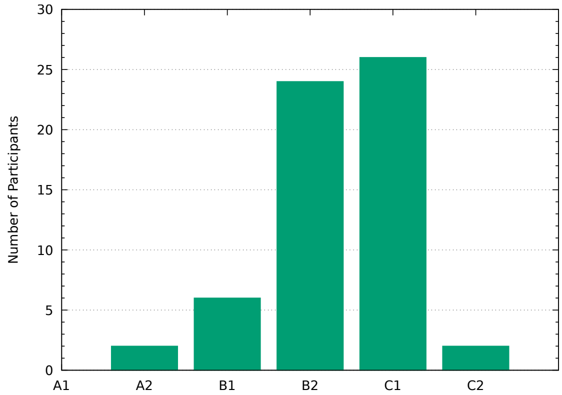


Figure 6: Our participants' CEFR level self-assessment

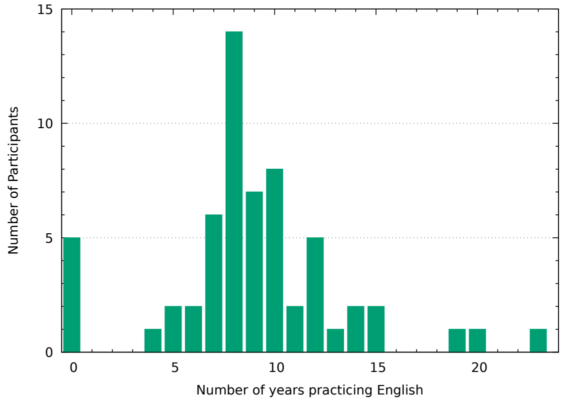


Figure 7: The number of years our participants have been practicing English

Figure 8: The frequency our participants have been practicing English

Readability index	T_1	T_2	T_3	T_4
Flesch reading ease	56.1	24.8	32	55.6
Gunning Fog	9.1	17.7	18.1	13.1
Flesch-Kincaid grade level	8.2	17.3	15.2	9.6
Coleman-Liau index	12	12	12	11
SMOG index	8.1	15.5	13.5	10.1
Automated readability index	7.9	17.4	15.5	9.7
Linsear Write formula	6.5	22.3	18.4	11.2

Table 2: Automated readability analysis of the four texts used for our C-tests. Scores are based on the online tool at http://www.readabilityformulas.com.

C-tests. Figure 9 shows the four texts T_1 to T_4 taken from the Brown corpus and the C-tests with the default gap scheme `DEF` we created from them for our user study. We have shortened each text to approximately 100 words and generated $n = 20$ gaps. In figure 10, we provide the results of our manipulation strategies `SEL` and `SIZE` with decreased ($\tau = 0.1$) and increased ($\tau = 0.5$) difficulty. Note that, we only show sentences that contain gaps; the beginning and end of each text is the same as in figure 9.

Table 2 reports readability scores for multiple common automated readability formulas. A Flesch reading ease score between 50–59 indicates *fairly difficult*, 30–49 *difficult*, and 0–29 *very difficult*. A Gunning Fog score of 9.1 indicates *fairly easy to read* and scores above 12 indicates *hard to read*. The remaining readability scores corresponding to grade levels.

The study of the St. Louis area's economic prospects prepared for the Construction Industry Joint Conference confirms and reinforces both the findings of the Metropolitan St. Louis Survey of 1957 and the easily observed picture of the Missouri-Illinois countryside. St. Louis si__ in t__ center o_ a relatively slow-growing a__ in so__ places stag____ mid-continent region . Slac_____ regional dem___ for St. Lo___ goods a__ services refl____ the reg___'s relative la__ of purch_____ power. N__ all St. Lo___ industries, o_ course, ha__ a market ar__ confined t_ the immediate neighborhood. But for those which do, the slow growth of the area has a retarding effect on the metropolitan core.

(a) C-test of T_1 with `DEF` gaps

Your invitation to write about Serge Prokofieff to honor his 70th Anniversary for the April issue of Sovietskaya Muzyka is accepted with pleasure, because I admire the music of Prokofieff; and with sober purpose, because the development of Prokofieff personifies, in many ways, the course of music in the Union of Soviet Socialist Republics. The Se___ Prokofieff wh__ we kn__ in t__ United Sta__ of Ame____ was g__, witty, merc_____, full o_ pranks a__ bonheur – a__ very cap____ as a profes______ musician. Th___ qualities ende____ him t_ both t__ musicians a__ the social-economic ha___ monde wh___ supported the concert world of the post-World War 1, era. Prokofieff's outlook as a composer-pianist-conductor in America was, indeed, brilliant.

(b) C-test of T_2 with `DEF` gaps

The superb intellectual and spiritual vitality of William James was never more evident than in his letters. Here w__ a man wi__ an enor____ gift f__ living a_ well a_ thinking. T_ both per____ and id___ he bro____ the sa__ delighted inte____, the sa__ open-minded relish f__ what w__ unique i_ each, t__ same discrim_______ sensibility a__ quicksilver intell______, the same gallantry of judgment. For this latest addition to the Great Letters Series, under the general editorship of Louis Kronenberger, Miss Hardwick has made a selection which admirably displays the variety of James's genius, not to mention the felicities of his style.

(c) C-test of T_3 with `DEF` gaps

Escalation unto death The nuclear war is already being fought, except that the bombs are not being dropped on enemy targets – not yet. It i_ being fou___, moreover, i_ fairly cl___ correspondence wi__ the predi______ of t__ soothsayers o_ the th___ factories. Th_ predicted escal_____, and escal_____ is wh__ we a__ getting. T__biggest nuc____ device t__ United Sta___ has expl____ measured so__ 15 megatons, although our B-52s are said to be carrying two 20-megaton bombs apiece. Some time ago, however, Mr. Khrushchev decided that when bigger bombs were made, the Soviet Union would make them.

(d) C-test of T_4 with `DEF` gaps

Figure 9: Standard C-tests of our user study

. . . The Serg_ Prokofieff who_ we kne_ in t__ United State_ of Americ_ was ga_, witty, mercuria_, full o_ pranks an_ bonheur – an_ very capabl_ as a professiona_ musician. Thes_ qualities endeare_ him t_ both t__ musicians an_ the social-economic haut_ monde whic_ supported. . .	. . . The S____ Prokofieff wh__ we kn__ in t__ United S_____ of A______ was ga_, witty, mercu____, full o_ pranks a__ bonheur – a__ very cap____ as a p__________ musician. T__ qualities end_____ him t_ both t__ musicians a__ the social-economic h____ monde wh___ supported. . .
(a) C-test of T_2 manipulated with SIZE for $\tau = 0.1$	(b) C-test of T_2 manipulated with SIZE for $\tau = 0.5$

. . . T__ Serge Proko_____ whom w_ kn__ i_ t__ Uni___ Sta___ o_ Ame___ w__ gay, witty, mercurial, fu__ o_ pranks and bonheur – a__ ve__ capable a__ a professional musician. These qualities endeared h__ t_ both t__ musicians a__ the social-economic haute monde which supported. . .	. . . The Se___ Prokofieff wh__ we kn__ in the United States of America was g__, wi___, merc_____, full of pra___ a__ bon____ – and very cap____ as a profes______ musi____. Th___ qual_____ ende____ h__ to bo__ the musi____ and the social-economic ha___ mo___ which supported. . .
(c) C-test of T_2 manipulated with SEL for $\tau = 0.1$	(d) C-test of T_2 manipulated with SEL for $\tau = 0.5$

. . . Here wa_ a man wit_ an enormou_ gift fo_ living a_ well a_ thinking. T_ both person_ and idea_ he brough_ the sa__ delighted interes_, the sa__ open-minded relish fo_ what wa_ unique i_ each, t__ same discriminatin_ sensibility an_ quicksilver intelligenc_, the same gallantry of judgment. . .	. . . Here w__ a man w___ an e_______ gift f__ living a_ well a_ thinking. T_ both per____ and id___ he bro____ the s___ delighted inte____, the s___ open-minded relish f__ what w__ unique i_ each, t__ same d___________ sensibility a__ quicksilver i__________, the same gallantry of judgment. . .
(e) C-test of T_3 manipulated with SIZE for $\tau = 0.1$	(f) C-test of T_3 manipulated with SIZE for $\tau = 0.5$

. . . Here w__ a m__ wi__ a_ enormous gift f__ liv___ a_ we__ a_ thinking. T_ both persons and ideas h_ bro____ t__ sa__ delighted interest, t__ sa__ open-minded relish f__ what w__ unique i_ each, t__ same discriminating sensibility and quicksilver intelligence, the same gallantry of judgment. . .	. . . He__ was a m__ with an enor____ gi__ for living as well as thin____. T_ bo__ per____ a__ id___ he brought the same deli_____ inte____, the same open-minded rel___ for wh__ was uni___ in ea__, the same discrim_______ sensi______ a__ quick______ intelligence, the same gallantry of judgment. . .
(g) C-test of T_3 manipulated with SEL for $\tau = 0.1$	(h) C-test of T_3 manipulated with SEL for $\tau = 0.5$

. . . It i_ being fough_, moreover, i_ fairly clos_ correspondence wit_ the prediction_ of t__ soothsayers o_ the thin_ factories. The_ predicted escalatio_, and escalatio_ is wha_ we ar_ getting. T__ biggest nuclea_ device t__ United State_ has explode_ measured som_ 15 megatons. . .	. . . It i_ being fou___, moreover, i_ fairly c____ correspondence w___ the p_________ of t__ soothsayers o_ the th___ factories. T___ predicted es_________, and es_________ is wh__ we a__ getting. T__ biggest nu_____ device t__ United Sta___ has expl____ measured s___ 15 megatons. . .
(i) C-test of T_4 manipulated with SIZE for $\tau = 0.1$	(j) C-test of T_4 manipulated with SIZE for $\tau = 0.5$

. . . I_ i_ be___ fou___, moreover, i_ fairly close correspondence wi__ t__ predictions o_ t__ soothsayers o_ t__ think factories. They predicted escalation, a__ escalation i_ wh__ w_ a__ getting. T__ big____ nuclear device t__ Uni___ States has exploded measured some 15 megatons. . .	. . . It is being fought, more____, in fai___ cl___ corresp_______ with the predi______ of the sooth______ of the th___ fact_____. Th__ pred_____ escal_____, and escal_____ is what w_ are get____. The big____ nuc____ dev___ the United States h__ expl____ meas____ some 15 megatons. . .
(k) C-test of T_4 manipulated with SEL for $\tau = 0.1$	(l) C-test of T_4 manipulated with SEL for $\tau = 0.5$

Figure 10: Manipulated C-tests of our user study

References

Lisa Marina Beinborn. 2016. *Predicting and manipulating the difficulty of text-completion exercises for language learning*. Ph.D. thesis, Technische Universität Darmstadt.

Timothy Dozat. 2016. Incorporating nesterov momentum into adam. In *ICLR Workshop*.

W. Nelson Francis. 1965. A standard corpus of edited present-day american english. *College English*, 26(4):267–273.

Shibamouli Lahiri. 2014. Complexity of Word Collocation Networks: A Preliminary Structural Analysis. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 96–105, Gothenburg, Sweden.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. RCV1: A New Benchmark Collection for Text Categorization Research. *Journal of Machine Learning Research*, 5(Apr):361–397.

Chapter 7

Constrained C-Test Generation via Mixed-Integer Programming

Constrained C-Test Generation via Mixed-Integer Programming

Ji-Ung Lee[1,†] **Marc E. Pfetsch**[2] **Iryna Gurevych**[1,†]
[1]Ubiquitous Knowledge Processing Lab, Department of Computer Science
[2] Research Group Optimization, Department of Mathematics
Technical University of Darmstadt, [†]Hessian AI

Abstract

This work proposes a novel method to generate C-Tests; a deviated form of cloze tests (a gap filling exercise) where only the last part of a word is turned into a gap. In contrast to previous works that only consider varying the gap size or gap placement to achieve locally optimal solutions, we propose a mixed-integer programming (MIP) approach. This allows us to consider gap size and placement simultaneously, achieving globally optimal solutions, and to directly integrate state-of-the-art models for gap difficulty prediction into the optimization problem. A user study with 40 participants across four C-Test generation strategies (including GPT-4) shows that our approach (`MIP`) significantly outperforms two of the baseline strategies (based on gap placement and GPT-4); and performs on-par with the third (based on gap size). Our analysis shows that GPT-4 still struggles to fulfill explicit constraints during generation and that `MIP` produces C-Tests that correlate best with the perceived difficulty. We publish our code, model, and collected data consisting of 32 English C-Tests with 20 gaps each (totaling 3,200 individual gap responses) under an open source license.[1]

1 Introduction

Learning a (second) language is one of the key factors that can directly impact a person's success in life. It enables them to participate in daily and social life and can even grant them new job opportunities. A crucial part of language learning is expanding the vocabulary by learning new words and their correct inflection. Gap filling exercises provide one possibility to consolidate new words and practice grammar rules (Oller Jr, 1973). Whereas most works focus on cloze tests (Taylor, 1953), a gap filling exercise where a whole word is turned into a gap, we focus on C-Tests, a deviated cloze test (Klein-Braley and Raatz, 1982). In contrast to cloze tests, C-Tests only remove the last part of a word, leaving the rest as a hint (cf. Fig. 1). This reduces the ambiguity of gap filling compared to cloze tests and requires learners to actively inflect words themselves. To provide sufficient context, the first and the last sentences remain free of gaps.

Despite above advantages, a major limitation of C-Tests is the prevalent static generation strategy that turns every second half of every second word into a gap. This impairs the usability of C-Tests for two reasons. First, the difficulty of a C-Test solely depends on the underlying text. Second, to suit a learner's curriculum, C-Tests need to be adapted post-generation; for instance, by manually placing gaps at words that consider a newly learned vocabulary. While novel C-Test generation strategies are necessary to tackle these issues, a key challenge is the large number of possible C-Tests that can be generated from a single text. For instance, placing m gaps among n words already results in $\binom{n}{m} = \frac{n!}{m!(n-m)!}$ possible C-Tests (cf. Appendix A.1). To reduce the complexity of the task, Lee et al. (2019) manually restrict the number of possible C-Tests and only vary the gap size with a static gap placement (or vice versa). Although they successfully generate C-Tests with varying difficulties using the same text, this discards a large number of C-Tests as potential solutions. Moreover, their strategies iteratively select the best gap size or placement which does not take the global interdependencies between gaps into account, leading to locally optimal solutions.

We instead propose to tackle C-Test generation as a mixed-integer programming (MIP) problem which results in three advantages over existing methods.[2] First, we can now make use of well-established solving methods that efficiently remove whole sets of unsuitable C-Tests while finding a provably optimal solution (see, e.g.,

[1]https://github.com/UKPLab/arxiv2024-constrained-ctest-generation

[2]We provide a primer on MIP in Appendix A.2.

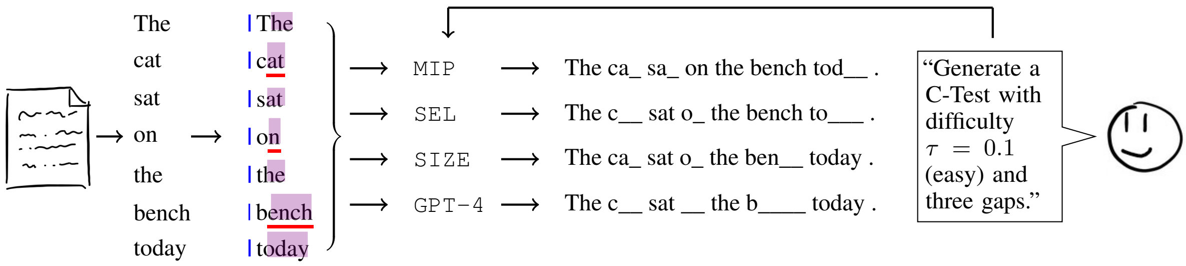


Figure 1: A simplified C-Test generation example. Colors indicate the gap sizes and words considered during generation. While `SIZE` (■) only varies the gap size with a static placement (every second word) and `SEL` (■) only the placement with a static gap size (the second half of a word, rounded up), `MIP` (■) considers all possible combinations. In contrast to `MIP`, purely neural approaches (`GPT-4`) provide no theoretical guarantee that all constraints are always satisfied. In this example, the word *on* is fully turned into a gap although the model correctly states in its response that words are only "partially deleted" in C-Tests (cf. Fig. 3 for the full prompt and response).

Schrijver 1986). This allows us to consider *all* possible C-Tests (instead of only a fraction) within a feasible run time of ~48.6 seconds.[3] Second, we can integrate trained gap difficulty prediction models into the optimization problem and solve the whole problem in an end-to-end manner (Bunel et al., 2018; Anderson et al., 2020). Third, in contrast to (purely neural) large language models (LLMs), the use of MIP provides a theoretical guarantee that the resulting C-Test always satisfies all constraints such as the number of gaps or their size (cf. Fig. 1). Together, these advantages allow teachers to directly generate C-Tests that suit their needs; eliminating the need to adapt them post-generation. Our contributions are:

- A novel generation method for C-Tests (`MIP`), that combines state-of-the-art models for gap difficulty prediction with constrained optimization methods.
- A user study with 40 participants that compares `MIP` against the gap size and placement generation strategies, as well as against C-Tests generated by GPT-4.
- Our data consisting of 32 C-Tests with 20 gaps each, their respective error rates, as well as perceived difficulties on a exercise-level.

2 Related Work

Various works have shown the usefulness of C-Tests in second language learning scenarios (Chapelle, 1994; Babaii and Ansary, 2001; Grotjahn, 2006; McKay, 2019). To select C-Tests that suit a learner's curriculum, it is necessary to predict their difficulty; i.e., proficient learners should receive more difficult C-Tests than inexperienced learners. This is done by either directly predicting difficulty of the whole C-Test (Settles et al., 2020; McCarthy et al., 2021), or—for a more fine-grained selection—by aggregating the individual gap difficulties (Lee et al., 2020). Predicting the difficulty of individual gaps is thus key, with past works investigating a wide range of features such as word frequency or readability scores across different models (Brown, 1989; Sigott, 1995; Eckes, 2011; Beinborn et al., 2014; Beinborn, 2016).

Instead of selecting suited C-Tests, one could also directly generate C-Tests with a specific difficulty. Towards this end, Lee et al. (2019) propose automated generation strategies for C-Tests based on a single text. Although they only vary the gap size or placement at once due to the large number of possible C-Tests, another advantage of deviating from the static generation strategy is a better quantification of a learner's proficiency (Cleary, 1988; Kamimoto, 1993; Laufer and Nation, 1999).

Finally, recent advances in LLMs have led to the emergence of new learning opportunities for students and teachers (Kohnke et al., 2023). Despite mixed results in essay scoring (Naismith et al., 2023; Yancey et al., 2023) and feedback generation (Duenas et al., 2023; Wang and Demszky, 2023), LLMs are easy to access and to use which makes them a tempting alternative to proprietary and expensive educational resources. As such, Xiao et al. (2023) investigate ChatGPT (Ouyang et al., 2022) to generate reading comprehension exercises and find that the model struggles to generate appropriate distractors (incorrect answers in multiple-choice questions). As C-Tests are solely

[3] In Appendix D.5, we devise methods that further reduce the run time to ~3.1 seconds.

based on deletion, generating them may be an easier task for LLMs than exercises that also require the generation of distractors. We thus include GPT-4 (OpenAI, 2023) as a baseline in our user study, but find that the model struggles to fulfill hard constraints such as the number of gaps.

In contrast, our method uses MIP—a more general form of integer linear programming (ILP)—which allows us to find globally optimal solutions (instead of locally optimal ones, cf. Lee et al. 2019) by relying upon constrained optimization methods which have been successfully applied across various NLP tasks (Roth and Yih, 2004; Barzilay and Lapata, 2006; Martins et al., 2009; Koo et al., 2010; Berant et al., 2011; Lin and Ng, 2021).

3 MIP Definition

Our goal is to generate a C-Test from a text $\mathcal{T}$ with a target difficulty τ and m gaps g. The gaps are selected from a set $\mathcal{G} \subset \mathcal{T}$ that denotes all words which can be turned into a gap (e.g., excluding first and last sentences). We further define $\tau \in [0, 1]$ to be the error rate computed over the whole C-Test:

$$\tau = \frac{1}{m}\sum_{i=1}^{m} \texttt{error}(g_i),$$

where g_i denotes the i-th gap of the C-Test. The function $\texttt{error}(\cdot)$ indicates if the i-th gap was filled-out correctly and returns a binary value (0 for correct and 1 for incorrect). Consequently, smaller values of τ relate to easier, and larger values to more difficult C-Tests. As the actual $\texttt{error}(\cdot)$ function is learner dependent and not known during generation, we approximate it using a gap difficulty prediction model $f_\theta : \mathbb{R}^k \mapsto [0, 1]$ with parameters θ that computes the error rate for each gap, represented as a k-dimensional, real number vector **x**.[4] We can now define the estimated difficulty $\hat{\tau}$ for any selection of m gaps $g \in \mathcal{G}$:

$$\hat{\tau} = \frac{1}{m}\sum_{i=1}^{m} f_\theta(g_i).$$

Given f_θ, any C-Test that has minimal distance between the estimated and the target difficulties is optimal. Hence, our optimization objective is:

$$\min \quad |\tau - \hat{\tau}|.$$

[4]From here on, we will refer to the gap error rate as the error rate and the C-Test error rate as the difficulty for clarity.

Gap placement. So far, $\hat{\tau}$ only includes gaps that have already been selected. To model the task of optimally placing them across all possible gaps $\mathcal{G}$ with $|\mathcal{G}| = n > m$, we now introduce binary decision variables:

$$\min_{b_i \in \{0,1\}} \quad |\tau - \frac{1}{m}\sum_{i=1}^{n} b_i\, f_\theta(g_i)|$$
$$\text{s.t.} \quad \sum_{i=1}^{n} b_i = m,$$

where b_i denotes a binary decision variable for a selected gap at the i-th word.[5] The constraint ensures that the resulting C-Test has exactly m gaps.

Gap size. In addition to the placement, each gap can assume a different size. We hence extend our objective with additional binary decision variables $s_{i,j}$ for the gap size where l_i denotes the length of the i-th word with $j \in \{1, \dots, l_i - 1\}$. Our final model comprising gap placement and gap size is then:

$$\min_{s_{i,j}, b_i \in \{0,1\}} \quad |\tau - \frac{1}{m}\sum_{i=1}^{n} b_i \sum_{j=1}^{l_i-1} s_{i,j} f_\theta(g_{i,j})| \quad (1)$$
$$\text{s.t.} \quad \sum_{i=1}^{n} b_i = m, \quad (2)$$
$$\sum_{j=1}^{l_i-1} s_{i,j} = 1, \quad (3)$$

where $g_{i,j}$ denotes a gap at the i-th word with size j, for all words $i \in \{1, \dots, n\}$. Our binary decision variables b_i and $s_{i,j}$ for the gap placement and size are constrained by Eq. (2) and Eq. (3), respectively.

Considerations. Analyzing our final model reveals three traits about the C-Test generation task and how we have defined it. First, we see that the number of possible gap placements is already very large with $\binom{n}{m}$; making it intractable to try out all combinations to identify an optimal C-Test with certainty. Second, despite the large number of possible combinations, the search space is still finite due to the discrete values of gap size and placement. Consequently, it is possible that two C-Tests are equally optimal with the same estimated difficulty $\hat{\tau}$. Third, we formulate the objective in a general way which does not include any learner-specific terms. Although this work does not investigate a

[5]Note, that i now ranges from 1 to n, with n denoting the number of all possible gaps and m the target number of gaps.

learner-specific adaptation, we note that the generation process can be tailored to a specific group of learners with a model trained on learner-specific data, or by adding learner-specific constraints (e.g., one that limits the gap size to a specific value).

4 Task Instantiation

Having defined our general optimization objective, we need to identify a suitable model f_θ for our evaluation study and define its constraints accordingly. As we consider f_θ including all its parameters in our objective, it needs to be small enough so that the whole optimization problem remains solvable in feasible time. We thus focus on the feature-based model proposed by Beinborn (2016) for predicting the gap error rate which achieves state-of-the-art performance. In addition to the 59 features utilized by Beinborn (2016), we further include two additional BERT-based features that have been shown to be helpful (Lee et al., 2020). A single gap $g_{i,j}$ is thus represented by a 61-dimensional vector $\mathbf{x}$, where each dimension k relates to a specific feature. As most of the features (51) do not depend on the gap size $s_{i,j}$ or placement b_i, we can model them as equality constraints for our model f_θ:

$$g_{i,j,k} = c_{i,j,k}, \tag{4}$$

where $c_{i,j,k}$ is the k-th pre-computed feature value for gap $g_{i,j,k}$.[6]

4.1 Gap Size Features

Overall, we identify six features that change depending on the gap size $s_{i,j}$. The first two are the BERT-based features that estimate the certainty that BERT correctly predicts the masked gap (Lee et al., 2020). For this, the authors use the probability of BERT predicting the correct solution ($k = 49$) and the entropy of the normalized vector of prediction probabilities for the top-50 candidates ($k = 50$). Next, we have three binary features that measure whether the gap occurs at a compound break, i.e., if the gap and the non-gap part are words on their own ($k = 56$), whether the non-gap part only consists of *th* ($k = 57$), and whether the gap begins at a syllable break ($k = 58$). Finally, we adapt the *WordLengthInCharacters* (Beinborn 2016, page 220) to model the varying gap size ($k = 59$), providing our model with a proper notion of gap size. We now describe our features that depend on the gap size in relation to our gap size decision variable $s_{i,j}$ by adding the following constraint to the MIP:

$$g_{i,j,k} = \mathbf{s}_i \cdot \mathrm{C}_{i,k}, \tag{5}$$

where $\mathbf{s}_i \in \mathbb{B}^{l_i-1}$ is the 1-hot vector (of length $l_i - 1$) representing the gap size decision variables (with 1 being at the j-th position) for gap g_i. The matrix $\mathrm{C}_{i,k} \in \mathbb{R}^{l_i-1\times k}$ represents all possible values $k \in \mathcal{U}$ can take for all possible gap sizes $j \in \{1, \ldots, l_i - 1\}$ at gap g_i with $\mathcal{U} = \{49, 50, 56, 57, 58, 59\}$; i.e., all our features that depend on the gap size.

4.2 Gap Placement Features

We identify four features that change depending on the gap placement. The first feature indicates if the word occurs somewhere else in the C-Test as a gap ($k = 51$). The second one represents the number of gaps in the same sentence ($k = 52$). The third and fourth features measure the number of preceding gaps in the C-Test ($k = 53$) and in the same sentence ($k = 54$). In contrast to the gap size features, all placement features depend on the placement of the other gaps. We thus need to model these dependencies into our constraints:

$$g_{i,j,51} = \max(\mathbf{b} \cdot \mathrm{V}), \tag{6}$$

$$g_{i,j,52} = \sum_{h \in \mathcal{S}_i} b_h, \tag{7}$$

$$g_{i,j,53} = \sum_{h=1}^{i-1} b_h, \tag{8}$$

$$g_{i,j,54} = \sum_{h \in \mathcal{S}_i, h<i} b_h, \tag{9}$$

for all $i, j \in \{1, \ldots, n\}$ where $\mathcal{S}_i$ denotes the set of all words in the sentence containing g_i. The vector $\mathbf{b}$ denotes all placement decision variables b_i and V the $n \times n$ matrix of binary values $v_{i,j}$ with:

$$v_{i,j} = \begin{cases} 1, & \text{if } w_i = w_j, \\ 0, & \text{otherwise,} \end{cases}$$

for all $i, j \in \{1, \ldots, n\}$ where $i \neq j$.

5 Gap Difficulty Model

With Eq. (1)–(9) defining our full optimization model, we focus on training a well-performing regression model f_θ that predicts the gap error rate. To ensure that the optimization model remains solvable in feasible time, we focus on small models with architectures that provide strong guarantees (Anderson et al., 2020).

[6] We provide a detailed description of all features in Appendix B.1 and an ablation study in Appendix C.3.2.

Dataset	Usage	# CT	# Gaps	μ GS	σ GS
ACL$_{2020\text{-train}}$	Train	69	1,480	2.80	1.30
ACL$_{2020\text{-test}}$	Dev	5	100	2.95	1.50
ACL$_{2019}$	Test	16	320	2.56	1.76

Table 1: Dataset statistics. # CT is the number of C-Tests per dataset. μ GS is the average gap size and σ GS the respective standard deviation. Each C-Test has 20 gaps.

Data. We use two datasets in total:

ACL-2020 Lee et al. (2020) provided us with their dataset for training our models.[7] It consists of 69 C-Tests we use for training, and 5 C-Tests (their test set) which we use as our development set. The data was collected by their university's language learning center from students taking language assessment tests and consists of gap error rates for C-Tests generated with the static generation strategy.

ACL-2019 Lee et al. (2019) published 16 C-Tests collected in their user study under a creative commons license. This is the only available data which contains gap error rates for C-Tests deviating from the static generation strategy, generated by `SEL` and `SIZE` (explained in the next section). We use this dataset as the test set to identify the best model for varying gap sizes and placements.

Table 1 shows the number of C-Tests, gaps (i.e., instances), and the data splits. We can see that the ACL-2019 data differs substantially from the ACL-2020 data as it has the smallest gap sizes on average (μ GS), but the largest standard deviation (σ GS).

Experimental setup. We consider four different model types for f_θ. Linear regression models (LR), support vector machines with a linear kernel (SVM), multi-layer perceptrons (MLP), and gradient boosted trees (XGB). We further include more recent models in our evaluation, namely, the base and large versions of BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), and DeBERTa (He et al., 2021). Each of the LR, SVM, MLP, and XGB models are trained to predict the gap error rate given the 61 respective features as the input. We evaluate three setups for the transformer-based models, namely, a masked-regression (MR) setup, a CLS-token setup, and a feature-enriched CLS-token setup (Appendix C.1). Following Beinborn et al. (2014), we use the root mean squared error (RMSE) and Pearson's correlation ρ to evaluate all models. We use an *Intel Core™ i5-8400 CPU* with 6 x 2.80GHz for training the LR, SVM, MLP, and XGB models and a single NVIDIA A100 with 80 GB for training the transformer-based models. We provide details on the training parameters and about hyperparameter tuning in Appendix C.2.

[7] We received permission to use the data for training and sharing our models (but not the data). We contacted authors of other works but without success due to proprietary restrictions.

Model	↓ RMSE	↑ Pearson's ρ
SVM$_{\text{Linear},c=0.01}$	1.846	−0.111
MLP$_{\text{ReLU}}$	1.113	0.099
MLP$_{\text{Linear}}$	1.681	0.085
LR	2.949	−0.385
XGB	**0.285**	**0.439**
BERT$_{\text{base}}$	0.311	0.279
BERT$_{\text{large}}$	0.319	0.174
RoBERTa$_{\text{base}}$	0.324	0.159
RoBERTa$_{\text{large}}$	0.324	0.050
DeBERTa$_{\text{v3-base}}$	0.311	0.245
DeBERTa$_{\text{v3-large}}$	0.308	0.259

Table 2: Root mean squared error (RMSE) and Pearson's correlation ρ for predicting the gap error rate on our test data. Overall, we find that XGB performs best, even outperforming large masked language models.

Results. Table 2 shows the results (averaged across ten runs) on the test data for the best performing models and transformer setup based on our hyperparameter tuning. Overall, we find that XGB performs best and is the only feature-based model that handles the varying gap sizes and placements well, as all other feature-based models show substantially worse scores. An analysis reveals that they may have overfitted on the training data, as they perform substantially better on the development data that closer matches the training data in terms of average gap size and variance. We further find a similar tendency for the feature-enriched transformer models (Table 7). We conjecture that XGB may be more robust to these changes due to its tree structure and the ensembling performed during training. Finally, we find a robust performance for the transformer-based models trained using only the CLS-token or in an MR manner; with a worse performance than the feature-based models on the development data, but a higher performance on the test data. Conducting an error analysis on the model with the best RMSE on the test data (DeBERTa$_{\text{v3-large}}$) reveals that the model performs better than XGB for small gap sizes (one and three), but worse for gap sizes equal or larger than four. We provide the full results and a detailed error analysis in Appendix C.3. We select the best performing model (XGB) for our user study.

6 User Study

To evaluate our optimization model, we conduct a user study where we ask our participants to complete C-Tests. Our main goal is to evaluate the following research hypothesis:

C-Tests that have been generated with our approach have a smaller distance $|\tau - \tau^*|$ (i.e., between the target difficulty τ and the observed difficulty τ^*) than C-Tests that have been generated with other approaches.

For a single C-Test, the observed difficulty is defined as:

$$\tau^* = \frac{1}{m \cdot r} \sum_{i=1}^{m} \sum_{j=1}^{r} \texttt{error}(g_{i,j}), \qquad (10)$$

where r is the number of individual responses for the i-th gap and m the total number of gaps in the C-Test. As before, $\texttt{error}(g_{i,j})$ returns 1 if $g_{i,j}$ was filled out incorrectly and 0 otherwise. We focus on the observed difficulty τ^*, as our goal is to evaluate the performance of the generation strategies and not the difficulty prediction model, which would require a different study setup (i.e., a study centered around quantifying $|\hat{\tau} - \tau^*|$).

6.1 Setup

Following Lee et al. (2019), we generate C-Tests with $m = 20$ gaps out of $n = 40$ possible words. We select two target difficulties $\tau \in \{0.1, 0.9\}$ as target difficulties that have been shown to be either easy ($\tau = 0.1$) or difficult ($\tau = 0.9$) for the baseline strategies to achieve (Lee et al., 2019).

Generation strategies. We compare our approach (`MIP`) against three generation strategies based on gap placement (`SEL`), size (`SIZE`) (Lee et al., 2019) and `GPT-4` (OpenAI, 2023).[8]

`SEL` The gap placement strategy first estimates all gap error rates for all $g \in \mathcal{G}$ in text $\mathcal{T}$ where the second half of a word is turned into a gap. It then iteratively selects the gaps with the smallest distance to τ; alternating between gaps that are easier and harder than τ.

`SIZE` The gap size strategy places gaps according to the static generation strategy. For each gap, the respective error rate is then computed and a character is removed or added iteratively until the target difficulty τ is reached. To increase computational efficiency, two separate models are trained on a reduced set of features which predict the relative change in difficulty for increasing or decreasing the gap size.

`GPT-4` Finally, we use GPT-4 to generate C-Tests. For a fair comparison, we only show the text passage containing $\mathcal{G}$ to the model (hence, no gaps are placed in the first and last sentence) and use five instances from the ACL-2019 dataset as few-shot examples in our prompt. We provide details on the few-shot example selection, C-Test generation, and the full final prompt in Appendix D.6.[9]

For `MIP`, we use Gurobi as a popular off-the-shelf solver (Gurobi Optimization, LLC 2023) to generate the C-Tests (with an average run time of 48.6 seconds). We provide a fine-grained analysis with respect to run time in Appendix D.1.2 and devise further improvements (reducing the average run time to 3.1 seconds) using different formulations of the optimization objective in Appendix D.5.

Text selection. We select text passages from four public domain books indexed at project Gutenberg.[10] Considering that all of our participants have an academic background, we select books with a high reading difficulty (Table 10). From each book, we randomly sample passages that contain at least three sentences and at least 40 words that can be turned into a gap (e.g., words that have at least two characters, do not contain any numbers, etc). We further avoid passages that contain dialogues. Data pre-processing (sentence splitting and tokenization) is done using NLTK's `sent_tokenize` and `word_tokenize` functions (Bird et al., 2009).

6.2 Design

Constructing the study with four texts $\mathcal{T}_1$–$\mathcal{T}_4$, two target difficulties $\tau \in \{0.1, 0.9\}$, and four genera-

[8] To select strong baselines, we reimplemented `SEL` and `SIZE` using the XGB model but found that the original models performed better. We provide results and detailed descriptions of all models in Appendix D.1.1.

[9] Note, that we cannot use the other datasets for GPT-4 due to the signed non-disclosure agreements regarding the data.

[10] `https://www.gutenberg.org/`

tion strategies results in 32 C-Tests (eight C-Tests for each generation strategy and text). To prevent participants from memorizing parts of the solution, each participant can only receive C-Tests generated from different texts, i.e., each participant is asked to solve four C-Tests.

Groups. To decide upon the groups (consisting of four C-Tests) to which our participants are assigned, we need to consider the respective configuration (i.e., which strategies and difficulties the group is composed of). Even a fixed ordering of the texts amounts to too many configurations to cover all—especially, as we also require multiple measurements for each C-Test to account for random effects such as our participants' individual performance. To obtain stable estimates that allow us to compare different generation strategies against each other, we construct our groups following a Latin Hypercube Experimental Design principle (McKay et al., 1979). This ensures that we select configurations with a minimal overlap between all possible combinations of text, strategy, and difficulty—with each text and strategy occurring once, and each target difficulty twice in all configurations (details in Appendix D.2.1).

Procedure. We implement and host a study interface using Flask[11] and SQLAlchemy[12]. On the landing page, participants are asked to enter a self-chosen study key of which we store the respective hash (so that we cannot guess a participant from the key). The key is only used in case a participant decides to access, change, or delete their data after the study. Upon registration, participants are informed about the purpose of the study, the collected data, and its use and are asked for their consent to participate in the study. Participants are further asked five questions about their English proficiency and are shown an example C-Test before being randomly assigned to one of the eight groups. After each of the four C-Tests, we ask our participants to provide a self-assessment of the perceived difficulty of the C-Test on a 5-point Likert scale and to estimate the number of gaps filled-out correctly. Figs. 8 to 10 show the questionnaire, the study interface, and the collected feedback to each C-Test, respectively.

[11] `https://flask.palletsprojects.com/`

[12] `https://www.sqlalchemy.org/`

$\lvert\tau - \tau^*\rvert$	μ	σ
`GPT-4`	0.45*	0.23
`MIP`	0.36	0.29
`SEL`	0.39*	0.27
`SIZE`	**0.34**°	0.29

Table 3: Average (μ) and standard deviation (σ) of $|\tau - \tau^*|$ for all generation strategies (lower is better). The Wald test (Wald, 1943) shows that `MIP` performs significantly better than `GPT-4` and `SEL` (*), but find no significant differences to `SIZE` (°).

6.3 Results

Overall, we recruited 40 volunteers for our study, resulting in five responses for each of the 32 C-Tests. All participants have at least B1 proficiency on the CEFR scale (Council of Europe, 2001) with a majority having C1 (16) or C2 (13) proficiency. Most of our participants are native German speakers (26) with the remainder distributed across 11 other languages.[13] On average, participants spent ∼4.5 minutes per C-Test and ∼17 minutes to finish all four C-Tests. Table 3 shows the average distance (μ) between the target difficulty τ and observed difficulty τ^* as well as the standard deviation (σ) and variance (σ^2).[14] While we can observe substantial differences between different generation strategies, we also see high standard deviations (larger than 0.2), making it difficult to interpret the results.

Statistical significance. To test if the observed differences with respect to the generation strategies are statistically significant, we fit a generalized additive mixed model (GAMM, Lin and Zhang 1999) on our collected data. This allows us to concurrently model our response variable (τ^*) on a continuous latent scale using multiple predictor variables expressed as a sum of smooth functions of covariates while accounting for random effects. Using the GAMM fitted on our data, we can now test for statistical significance between different C-Test generation strategies using the Wald test (Wald, 1943). Overall, we find that all strategies significantly outperform `GPT-4` which has the highest $|\tau - \tau^*|$ (cf. Table 3). In addition, `MIP` and `SIZE` significantly outperform `SEL`. Although `MIP` performs slightly worse than `SIZE`, we find that the differences are not statistically significant. Interest-

[13] We provide detailed statistics in Appendix D.3.1.

[14] We compute the observed difficulty as the C-Test error rate, i.e., the fraction of incorrect gaps over all 20 gaps.

ingly, we find that C-Tests generated using `GPT-4` show a reverse tendency for τ^*, displaying a significantly higher difficulty for $\tau = 0.1$ than for $\tau = 0.9$ ($p < 0.001$). In other words, our participants made significantly more errors on `GPT-4`-generated C-Tests that were supposed to be easier and vice versa.[15]

τ	GPT-4	MIP	SEL	SIZE
0.1	4.3±0.56	**2.6**±0.97	3.5±0.74	3.0±1.16
0.9	4.0±0.56	3.9±0.70	**4.2**±0.60	3.8±0.83

Table 4: Average perceived difficulty on a 5-point Likert scale between too easy (1) and too hard (5).

Perceived difficulty. Table 4 shows the average estimates of the perceived C-Test difficulty on a 5-point Likert scale (Likert, 1932) between *too easy* (1) and *too hard* (5). Overall, we find the largest difference in perceived difficulty for `MIP` (1.3; i.e., more than one rating), indicating that `MIP` generated C-Tests that were perceived substantially easier (or harder) for $\tau = 0.1$ ($\tau = 0.9$). Whereas we find similar tendencies for `SEL` and `SIZE`, C-Tests generated with `GPT-4` again show a reversed tendency for the perceived difficulty (i.e., our participants found presumably easy C-Tests harder than the difficult ones). To check for statistical significance, we fit a second GAMM for ordinal data and find that `MIP` is again significantly better than `SEL` and `GPT-4` with no significant difference to `SIZE`.[16]

6.4 Error Analysis

`MIP` vs. `SIZE`. Analyzing the C-Tests generated by `MIP` reveals that the generation strategy may struggle with assessing interdependencies between gaps. For instance, we find that the C-Test generated from $\mathcal{T}_1$ with $\tau = 0.1$ contains multiple successive gaps (Fig. 2, left). This substantially increases the difficulty compared to the `SIZE` strategy that only places gaps at every second word (Fig. 2, right). Intuitively, gaps that occur in succession should be harder to fill out—e.g., consider a C-Test where a single sentence only contains gaps versus a C-Test where the gaps are evenly distributed. However, we find no patterns with respect to the length and occurrences of successive gaps for C-Tests with varying difficulties (Table 16). We identify three causes for this shortcoming and discuss potential solutions to be addressed in future work. Each cause can either be attributed to the difficulty prediction model f_θ (i.e., the XGB model) or to the optimization model (i.e., `MIP`). First, f_θ has only been trained on C-Tests generated by the static strategy which leads to a lack of successive gaps during training; despite the decent performance on the test data with different gap sizes and placements. This work alleviates this issue by providing data with varying gap sizes and placements for training. Second, f_θ only uses features that implicitly capture the interdependency such as the number of gaps in a sentence. This could be tackled by explicitly modeling interdependencies; e.g., with a binary feature that indicates if the previous word is a gap. Finally, `MIP` does not specifically model interdependencies. One way to better capture interdependencies could be to introduce a weighting term in our objective (Eq. (1)) and increase the estimated gap error rate according to the number of successive gaps.

	τ	$\mathcal{T}_1$	$\mathcal{T}_2$	$\mathcal{T}_3$	$\mathcal{T}_4$
# Gaps	0.1	20^5	20	20^2	20^2
	0.9	29	25	37	22
μ Size	0.1	3.4	4.1	3.6	3.7
	0.9	3.45	3.75	2.9	3.4

Table 5: Number of gaps and average size for C-Tests generated by `GPT-4`. Superscripts denote the number of required regenerations to obtain at least 20 gaps.

Shortcomings of `GPT-4`. The increasing use of GPT-4 (and ChatGPT) in education was a key reason to include it in the study (Zhang et al., 2023a), making it even more concerning that the model performed worst. Analyzing the C-Tests generated by `GPT-4` reveals that gaps are frequently clustered at the beginning for $\tau = 0.9$. Table 5 shows the number of generated gaps per text (# Gaps) and their average size (μ Size) after pruning to 20 gaps. We find that the model generates substantially more gaps for $\tau = 0.9$, which shows that it lacks a notion of gap-level difficulty and simply adds more gaps to increase the difficulty. Moreover, the model generates convincing (but incorrect) explanations along with the exercise (cf. Appendix D.4), which could be especially harmful in self-directed learning scenarios with only GPT-4 as the tutor. This highlights the importance of approaches that better

[15] Detailed statistics and the formular of the GAMM are provided in Appendix D.3.2.

[16] We provide details on the second GAMM and box plots for the perceived difficulties in Appendix D.3.3.

```
...As little_ as the act of birth comes
into consideration in the who__
proces_ a__ procedur_ of heredit_,
ju__ as littl_ is "being-conscious"
oppose_ t_ the instinctiv_ in a__
d_______ sen__; the grea___ p___ of
the consciou_ thi_____ of a
philosophe_ i_ secretly influenced by
his instincts...
```

(a) C-test of $\mathcal{T}_1$ generated with MIP for $\tau = 0.1$

```
...As lit___ as t__ act o_ birth
com__ into consider_____ in t__ whole
pro_____ and proc______ of here_____,
just a_ little i_ "being-conscious"
opposed to t__ instinctive i_ any
deci_____ sense; t__ greater pa__ of
t__ conscious thin____ of a
philo______ is secr____ influneced by
his instincts...
```

(b) C-test of $\mathcal{T}_1$ generated with SIZE for $\tau = 0.1$

Figure 2: MIP vs SIZE. Colored squares indicate the gap error rates (0.0 1.0)

control the output of LLMs (Zhang et al., 2023b).

Tuning the prompt. To better understand the extent of above issues, we explored more sophisticated prompting strategies (after the study) with the goal to improve GPT-4's notion of C-Test difficulty. First, we asked the model to generate multiple C-Tests at once with both target difficulties. Second, we provided corrective feedback while asking it to increase or decrease the difficulty. First and foremost, we find that GPT-4 has undergone substantial changes in the meantime, as it provides very different responses. Most notably, the model now prioritizes a modification of the gap size to increase or decrease the C-Test difficulty. Despite this improvement, the key issues remain as there are still instances where whole words are turned into gaps or where the model decreases the gap size when prompted to increase the difficulty. Interestingly, we find that GPT-3.5 provides responses that are similar to those during study development; with a model that mostly aims to control the difficulty with the number of placed gaps. Overall, we conclude that both models still struggle to follow hard constraints and moreover, that they lack an inherent notion of C-Test difficulty.

7 Conclusion

This work proposes a first constrained generation strategy for C-Tests, a type of gap filling exercise. We provide a general MIP formulation for C-Test generation and specify the optimization problem for a state-of-the-art model. A user study with 40 participants across four generation strategies shows that our approach significantly outperforms two baselines and performs on-par with the third. Our approach further generates C-Tests that resonates best with our participants in terms of perceived difficulty. This could be promising to investigate in future work, as the perceived difficulty can substantially impact a learner's motivation. Finally, our analysis reveals two further research directions for future work; modeling interdependencies between gaps and making LLMs usable for educational purposes by better controlling their output.

8 Limitations

While our proposed approach performed reasonably well and is the only one that provides theoretical guarantees to generate C-Tests that fulfill all hard constraints, it suffers from three limitations.

Scaling. First, it cannot scale indefinitely to larger models as the consumption of computational resources scales exponentially with the increasing model size in the worst case (cf. Appendix A.2). Moreover, existing solving methods do not transfer well to GPUs due to a limited parallelization. In comparison, the gap size strategy scales linearly as it is only limited by the number (and length) of the words considered as potential gaps (but does not provide any theoretical guarantees as its solutions are approximate). We explore one possible way to alleviate the scaling issues to some extent by investigating other, mathematically equal formulations of the optimization objective in Appendix D.5 and find that we can substantially reduce the run time. Another, fundamentally different research direction could be to only include the upper layers of a model in the MIP; considering the output of lower layers as fixed input features (as is done in this work). This remains to be investigated in future work.

Other architectures. Second, more complex activation functions such as Gaussian Error Linear Units (GELU) are increasingly being used in recent models, but their respective MIP formulations do not provide any guarantees so far. This remains an open research question.

GPT-4. Finally, we used GPT-4 instead of an open source model such as BLOOM (BigScience Workshop et al., 2023), considering the accessibility of models for teachers who are interested in using LLMs to generate exercises. Following this thought, we tried to keep the prompt as simple as possible and tuned it until the model was capable of generating gaps properly (interestingly, asking the model to generate the correct number of blanks resulted in faulty responses). An evaluation with teachers to gain more insights into how domain experts interact with different C-Test generation systems is ongoing. This also includes a prompting-specific training with methods such as chain-of-thought prompting (Wei et al., 2022) and role-play prompting (Kong et al., 2024).

9 Ethical Considerations

Data collection. The study fulfills all conditions of our university's guidelines for ethical research and has been approved by a spokesperson of the ethics committee of our university. To ensure a GDPR-conform data collection, we do not collect any personal data of our participants. Before participation, every participant is informed about the collected data, its usage, and instructed on how their data can be accessed, edited, deleted post-study. Participation is only possible upon consent; if not provided, any collected data such as the hash of the study key are immediately deleted. All our participants were volunteers who participated out of self-interest and received no compensation. Although this made the recruitment of participants more difficult, we conjectured that this would result in more motivated participants (and responses of higher quality), in contrast to setups where their main motivation is some form of compensation (money, course credit, or something else). All data is anonymized for publication.

Risks. Our findings with respect to GPT-4 show that although the model struggles with fulfilling hard constraints, it can still generate convincing (but misleading) explanations. This emphasizes that the use of LLMs in the educational domain requires careful consideration, especially in the context of self-directed learning where no teacher is present. Finally, we note that the models we investigate in this research are primarily developed for English. While we provide a general formulation of the optimization problem in Section 3, this requires further adaptation to language-specific models which may be difficult to obtain especially for endangered languages. However, we note that the considerable performance of the XGB model with a rather small training dataset could provide a chance; easing the adaptation of our approach to other languages.

Acknowledgments

We thank Hendrik Schuff for the insightful discussions about the user study setup and Max Glockner for the interesting discussions on the general storyline. We further thank Jiahui Geng, Jan-Christoph Klie, Toru Sasaki, and Martin Tutek for their helpful feedback on the paper draft. Finally, we thank our anonymous reviewers who provided insightful feedback and engaged in discussions. This work has been funded by the LOEWE Distinguished Chair "Ubiquitous Knowledge Processing", LOEWE initiative, Hesse, Germany (Grant Number: LOEWE/4a//519/05/00.002(0002)/81) and the German Research Foundation (DFG) for the project "Globally Optimal Neural Network Training" within the SPP 2298.

References

Tobias Achterberg, Robert E Bixby, Zonghao Gu, Edward Rothberg, and Dieter Weninger. 2020. Presolve reductions in mixed integer programming. *INFORMS Journal on Computing*, 32(2):473–506.

Ross Anderson, Joey Huchette, Will Ma, Christian Tjandraatmadja, and Juan Pablo Vielma. 2020. Strong mixed-integer programming formulations for trained neural networks. *Mathematical Programming*, 183(1-2):3–39.

Esmat Babaii and Hasan Ansary. 2001. The C-test: a valid operationalization of reduced redundancy principle? *System*, 29(2):209–219.

Regina Barzilay and Mirella Lapata. 2006. Aggregation via set partitioning for natural language generation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 359–366, New York City, USA. Association for Computational Linguistics.

Amitabh Basu, Michele Conforti, Marco Di Summa, and Hongyi Jiang. 2023. Complexity of branch-and-bound and cutting planes in mixed-integer optimization. *Mathematical Programming*, 198(1):787–810.

Lisa Beinborn, Torsten Zesch, and Iryna Gurevych. 2014. Predicting the Difficulty of Language Proficiency Tests. *Transactions of the Association for Computational Linguistics*, 2:517–530.

Lisa Marina Beinborn. 2016. *Predicting and manipulating the difficulty of text-completion exercises for language learning*. Ph.D. thesis, Technische Universität Darmstadt.

Jonathan Berant, Ido Dagan, and Jacob Goldberger. 2011. Global learning of typed entailment rules. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 610–619, Portland, Oregon, USA. Association for Computational Linguistics.

BigScience Workshop, :, Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurençon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa, Alham Fikri Aji, Amit Alfassy, Anna Rogers, Ariel Kreisberg Nitzav, Canwen Xu, Chenghao Mou, Chris Emezue, Christopher Klamm, Colin Leong, Daniel van Strien, David Ifeoluwa Adelani, Dragomir Radev, Eduardo González Ponferrada, Efrat Levkovizh, Ethan Kim, Eyal Bar Natan, Francesco De Toni, Gérard Dupont, Germán Kruszewski, Giada Pistilli, Hady Elsahar, Hamza Benyamina, Hieu Tran, Ian Yu, Idris Abdulmumin, Isaac Johnson, Itziar Gonzalez-Dios, Javier de la Rosa, Jenny Chim, Jesse Dodge, Jian Zhu, Jonathan Chang, Jörg Frohberg, Joseph Tobing, Joydeep Bhattacharjee, Khalid Almubarak, Kimbo Chen, Kyle Lo, Leandro Von Werra, Leon Weber, Long Phan, Loubna Ben allal, Ludovic Tanguy, Manan Dey, Manuel Romero Muñoz, Maraim Masoud, María Grandury, Mario Šaško, Max Huang, Maximin Coavoux, Mayank Singh, Mike Tian-Jian Jiang, Minh Chien Vu, Mohammad A. Jauhar, Mustafa Ghaleb, Nishant Subramani, Nora Kassner, Nurulaqilla Khamis, Olivier Nguyen, Omar Espejel, Ona de Gibert, Paulo Villegas, Peter Henderson, Pierre Colombo, Priscilla Amuok, Quentin Lhoest, Rheza Harliman, Rishi Bommasani, Roberto Luis López, Rui Ribeiro, Salomey Osei, Sampo Pyysalo, Sebastian Nagel, Shamik Bose, Shamsuddeen Hassan Muhammad, Shanya Sharma, Shayne Longpre, Somaieh Nikpoor, Stanislav Silberberg, Suhas Pai, Sydney Zink, Tiago Timponi Torrent, Timo Schick, Tristan Thrush, Valentin Danchev, Vassilina Nikoulina, Veronika Laippala, Violette Lepercq, Vrinda Prabhu, Zaid Alyafeai, Zeerak Talat, Arun Raja, Benjamin Heinzerling, Chenglei Si, Davut Emre Taşar, Elizabeth Salesky, Sabrina J. Mielke, Wilson Y. Lee, Abheesht Sharma, Andrea Santilli, Antoine Chaffin, Arnaud Stiegler, Debajyoti Datta, Eliza Szczechla, Gunjan Chhablani, Han Wang, Harshit Pandey, Hendrik Strobelt, Jason Alan Fries, Jos Rozen, Leo Gao, Lintang Sutawika, M Saiful Bari, Maged S. Al-shaibani, Matteo Manica, Nihal Nayak, Ryan Teehan, Samuel Albanie, Sheng Shen, Srulik Ben-David, Stephen H. Bach, Taewoon Kim, Tali Bers, Thibault Fevry, Trishala Neeraj, Urmish Thakker, Vikas Raunak, Xiangru Tang, Zheng-Xin Yong, Zhiqing Sun, Shaked Brody, Yallow Uri, Hadar Tojarieh, Adam Roberts, Hyung Won Chung, Jaesung Tae, Jason Phang, Ofir Press, Conglong Li, Deepak Narayanan, Hatim Bourfoune, Jared Casper, Jeff Rasley, Max Ryabinin, Mayank Mishra, Minjia Zhang, Mohammad Shoeybi, Myriam Peyrounette, Nicolas Patry, Nouamane Tazi, Omar Sanseviero, Patrick von Platen, Pierre Cornette, Pierre François Lavallée, Rémi Lacroix, Samyam Rajbhandari, Sanchit Gandhi, Shaden Smith, Stéphane Requena, Suraj Patil, Tim Dettmers, Ahmed Baruwa, Amanpreet Singh, Anastasia Cheveleva, Anne-Laure Ligozat, Arjun Subramonian, Aurélie Névéol, Charles Lovering, Dan Garrette, Deepak Tunuguntla, Ehud Reiter, Ekaterina Taktasheva, Ekaterina Voloshina, Eli Bogdanov, Genta Indra Winata, Hailey Schoelkopf, Jan-Christoph Kalo, Jekaterina Novikova, Jessica Zosa Forde, Jordan Clive, Jungo Kasai, Ken Kawamura, Liam Hazan, Marine Carpuat, Miruna Clinciu, Najoung Kim, Newton Cheng, Oleg Serikov, Omer Antverg, Oskar van der Wal, Rui Zhang, Ruochen Zhang, Sebastian Gehrmann, Shachar Mirkin, Shani Pais, Tatiana Shavrina, Thomas Scialom, Tian Yun, Tomasz Limisiewicz, Verena Rieser, Vitaly Protasov, Vladislav Mikhailov, Yada Pruksachatkun, Yonatan Belinkov, Zachary Bamberger, Zdeněk Kasner, Alice Rueda, Amanda Pestana, Amir Feizpour, Ammar Khan, Amy Faranak, Ana Santos, Anthony Hevia, Antigona Unldreaj, Arash Aghagol, Arezoo Abdollahi, Aycha Tammour, Azadeh HajiHosseini, Bahareh Behroozi, Benjamin Ajibade, Bharat Saxena, Carlos Muñoz Ferrandis, Daniel McDuff, Danish Contractor, David Lansky, Davis David, Douwe Kiela, Duong A. Nguyen, Edward Tan, Emi Baylor, Ezinwanne Ozoani, Fatima Mirza, Frankline Ononiwu, Habib Rezanejad, Hessie Jones, Indrani Bhattacharya, Irene Solaiman, Irina Sedenko, Isar Nejadgholi, Jesse Passmore, Josh Seltzer, Julio Bonis Sanz, Livia Dutra, Mairon Samagaio, Maraim Elbadri, Margot Mieskes, Marissa Gerchick, Martha Akinlolu, Michael McKenna, Mike Qiu, Muhammed Ghauri, Mykola Burynok, Nafis Abrar, Nazneen Rajani, Nour Elkott, Nour Fahmy, Olanrewaju Samuel, Ran An, Rasmus Kromann, Ryan Hao, Samira Alizadeh, Sarmad Shubber, Silas Wang, Sourav Roy, Sylvain Viguier, Thanh Le, Tobi Oyebade, Trieu Le, Yoyo Yang, Zach Nguyen, Abhinav Ramesh Kashyap, Alfredo Palasciano, Alison Callahan, Anima Shukla, Antonio Miranda-Escalada, Ayush Singh, Benjamin Beilharz, Bo Wang, Caio Brito, Chenxi Zhou, Chirag Jain, Chuxin Xu, Clémentine Fourrier, Daniel León Periñán, Daniel Molano, Dian Yu, Enrique Manjavacas, Fabio Barth, Florian Fuhrimann, Gabriel Altay, Giyaseddin Bayrak, Gully Burns, Helena U. Vrabec, Imane Bello, Ishani Dash, Jihyun Kang, John Giorgi, Jonas Golde, Jose David Posada, Karthik Rangasai Sivaraman, Lokesh Bulchandani, Lu Liu, Luisa Shinzato, Madeleine Hahn de Bykhovetz, Maiko Takeuchi, Marc Pàmies, Maria A Castillo, Marianna Nezhurina, Mario Sänger, Matthias Samwald, Michael Cullan, Michael Weinberg, Michiel De Wolf, Mina Mihaljcic, Minna Liu, Moritz Freidank, Myungsun Kang, Natasha Seelam, Nathan Dahlberg, Nicholas Michio Broad, Nikolaus Muellner, Pascale Fung, Patrick Haller, Ramya Chandrasekhar, Renata Eisenberg, Robert Martin, Rodrigo Canalli, Rosaline Su, Ruisi Su, Samuel Cahyawijaya, Samuele Garda, Shlok S Deshmukh, Shubhanshu Mishra, Sid Kiblawi, Simon Ott, Sinee Sang-aroonsiri, Srishti Kumar, Stefan Schweter, Sushil Bharati, Tanmay Laud, Théo Gigant, Tomoya Kainuma, Wojciech Kusa, Yanis Labrak, Yash Shailesh Bajaj, Yash Venkatraman, Yifan Xu, Yingxin Xu, Yu Xu, Zhe Tan, Zhongli Xie, Zifan Ye, Mathilde Bras, Younes Belkada, and Thomas Wolf. 2023. Bloom: A 176b-parameter open-access multilingual language model.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

James Dean Brown. 1989. Cloze item difficulty. *Japan Association for Language Teaching (JALT)*, 11(1):46–67.

James L Bruning and Buddy L Kintz. 1987. *Computational handbook of statistics*. Scott, Foresman & Co.

Rudy Bunel, Ilker Turkaslan, Philip H.S. Torr, Pushmeet Kohli, and M. Pawan Kumar. 2018. A Unified View of Piecewise Linear Neural Network Verification. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, page 4795–4804, Red Hook, NY, USA. Curran Associates Inc.

C. A. Chapelle. 1994. Are C-tests valid measures for L2 vocabulary research? *Second Language Research*, 10(2):157–187.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIG-KDD international conference on knowledge discovery and data mining*, pages 785–794.

Christopher Cleary. 1988. The c-test in english: left-hand deletions. *Regional Language Centre (RELC)*, 19(2):26–35.

Council of Europe. 2001. *Common European framework of reference for languages: Learning, teaching, assessment*. Cambridge University Press.

Sanjeeb Dash. 2002. An exponential lower bound on the length of some classes of branch-and-cut proofs. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 145–160. Springer.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

George Duenas, Sergio Jimenez, and Geral Mateus Ferro. 2023. You've Got a Friend in ... a Language Model? A Comparison of Explanations of Multiple-Choice Items of Reading Comprehension between ChatGPT and Humans. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, pages 372–381, Toronto, Canada. Association for Computational Linguistics.

Thomas Eckes. 2011. Item banking for c-tests: A polytomous rasch modeling approach. *Psychological Test and Assessment Modeling*, 53(4):414.

Ralph E. Gomory. 1960. An algorithm for the mixed integer problem. *Report No. P-1885, The Rand Corporation, Santa Monica, CA*.

Rüdiger Grotjahn. 2006. *Der C-Test: Theorie, Empirie, Anwendungen / The C-Test: Theory, Empirical Research, Applications*. Peter Lang Verlag, Berlin, Germany.

Gurobi Optimization, LLC. 2023. Gurobi Optimizer Reference Manual.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. DeBERTa: Decoding-enhanced BERT with Disentangled Attention. In *International Conference on Learning Representations*, pages 1–21.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.

Tadamitsu Kamimoto. 1993. Tailoring the test to fit the students : Improvement of the c-test through classical item analysis. *Language Laboratory*, 30:47–61.

Christine Klein-Braley and Ulrich Raatz. 1982. Der C-Test: ein neuer Ansatz zur Messung allgemeiner Sprachbeherrschung. *AKS-Rundbrief*, 4:23–37.

Lucas Kohnke, Benjamin Luke Moorhouse, and Di Zou. 2023. ChatGPT for Language Teaching and Learning. *Regional Language Centre (RELC)*, 54(2):537–550.

Aobo Kong, Shiwan Zhao, Hao Chen, Qicheng Li, Yong Qin, Ruiqi Sun, Xin Zhou, Enzhi Wang, and Xiaohang Dong. 2024. Better Zero-Shot Reasoning with Role-Play Prompting. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, page (to appear), Mexico City, Mexico.

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1288–1298, Cambridge, MA. Association for Computational Linguistics.

AH Land and AG Doig. 1960. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520.

Batia Laufer and Paul Nation. 1999. A vocabulary-size test of controlled productive ability. *Language Testing*, 16(1):33–51.

Ji-Ung Lee, Christian M. Meyer, and Iryna Gurevych. 2020. Empowering Active Learning to Jointly Optimize System and User Demands. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4233–4247, Online. Association for Computational Linguistics.

Ji-Ung Lee, Erik Schwan, and Christian M. Meyer. 2019. Manipulating the difficulty of C-tests. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 360–370, Florence, Italy. Association for Computational Linguistics.

Rensis Likert. 1932. A technique for the measurement of attitudes. *Archives of psychology*, 22(140):5–55.

Ruixi Lin and Hwee Tou Ng. 2021. System combination for grammatical error correction based on integer programming. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, pages 824–829, Held Online. INCOMA Ltd.

Xihong Lin and Daowen Zhang. 1999. Inference in generalized additive mixed models by using smoothing splines. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 61(2):381–400.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*, pages 1–18.

André F. T. Martins, Noah A. Smith, and Eric P. Xing. 2009. Polyhedral outer approximations with application to natural language parsing. In *Proceedings of the 26th Annual International Conference on Machine Learning*, page 713–720, New York, NY, USA. Association for Computing Machinery.

Arya D. McCarthy, Kevin P. Yancey, Geoffrey T. LaFlair, Jesse Egbert, Manqian Liao, and Burr Settles. 2021. Jump-starting item parameters for adaptive language tests. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 883–899, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

M. D. McKay, R. J. Beckman, and W. J. Conover. 1979. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245.

Todd McKay. 2019. *More on the validity and reliability of C-test scores: a meta-analysis of C-test studies*. Ph.D. thesis, Georgetown University.

Ben Naismith, Phoebe Mulcaire, and Jill Burstein. 2023. Automated evaluation of written discourse coherence using GPT-4. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, pages 394–403, Toronto, Canada. Association for Computational Linguistics.

Jorge Nocedal and Stephen J Wright. 1999. *Numerical optimization*. Springer.

John W Oller Jr. 1973. Cloze tests of second language proficiency and what they measure. *Language learning*, 23(1):105–118.

OpenAI. 2023. GPT-4 Technical Report.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:1–12.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

R Core Team. 2022. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Dan Roth and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*, pages 1–8, Boston, Massachusetts, USA. Association for Computational Linguistics.

Alexander Schrijver. 1986. *Theory of linear and integer programming*. John Wiley & Sons.

Burr Settles, Geoffrey T. LaFlair, and Masato Hagiwara. 2020. Machine Learning–Driven Language Assessment. *Transactions of the Association for Computational Linguistics*, 8:247–263.

Günther Sigott. 1995. The C-Test: Some Factors of Difficulty. *AAA: Arbeiten aus Anglistik und Amerikanistik*, 20(1):43–53.

Wilson L. Taylor. 1953. "Cloze Procedure": A New Tool for Measuring Readability. *Journalism Bulletin*, 30(4):415–433.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Marcos Treviso, Ji-Ung Lee, Tianchu Ji, Betty van Aken, Qingqing Cao, Manuel R. Ciosici, Michael Hassid, Kenneth Heafield, Sara Hooker, Colin Raffel, Pedro H. Martins, André F. T. Martins, Jessica Zosa Forde, Peter Milder, Edwin Simpson, Noam Slonim, Jesse Dodge, Emma Strubell, Niranjan Balasubramanian, Leon Derczynski, Iryna Gurevych, and Roy Schwartz. 2023. Efficient Methods for Natural Language Processing: A Survey. *Transactions of the Association for Computational Linguistics*, 11:826–860.

Abraham Wald. 1943. Tests of statistical hypotheses concerning several parameters when the number of observations is large. *Transactions of the American Mathematical society*, 54(3):426–482.

Rose Wang and Dorottya Demszky. 2023. Is ChatGPT a good teacher coach? measuring zero-shot performance for scoring and providing actionable insights on classroom instruction. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, pages 626–667, Toronto, Canada. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Laurence A Wolsey and George L Nemhauser. 1998. *Integer and combinatorial optimization*, volume 55. John Wiley & Sons.

Simon N Wood, Natalya Pya, and Benjamin Säfken. 2016. Smoothing parameter and model selection for general smooth models. *Journal of the American Statistical Association*, 111(516):1548–1563.

Changrong Xiao, Sean Xin Xu, Kunpeng Zhang, Yufang Wang, and Lei Xia. 2023. Evaluating reading comprehension exercises generated by LLMs: A showcase of ChatGPT in education applications. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, pages 610–625, Toronto, Canada. Association for Computational Linguistics.

Kevin P. Yancey, Geoffrey Laflair, Anthony Verardi, and Jill Burstein. 2023. Rating short L2 essays on the CEFR scale with GPT-4. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, pages 576–584, Toronto, Canada. Association for Computational Linguistics.

Amir Zeldes. 2017. The GUM Corpus: Creating Multilayer Resources in the Classroom. *Language Resources and Evaluation*, 51(3):581–612.

Chaoning Zhang, Chenshuang Zhang, Chenghao Li, Yu Qiao, Sheng Zheng, Sumit Kumar Dam, Mengchun Zhang, Jung Uk Kim, Seong Tae Kim, Jinwoo Choi, Gyeong-Moon Park, Sung-Ho Bae, Lik-Hang Lee, Pan Hui, In So Kweon, and Choong Seon Hong. 2023a. One Small Step for Generative AI, One Giant Leap for AGI: A Complete Survey on ChatGPT in AIGC Era. *arXiv preprint 2304.06488*.

Hanqing Zhang, Haolin Song, Shaoyu Li, Ming Zhou, and Dawei Song. 2023b. A Survey of Controllable Text Generation Using Transformer-Based Pre-Trained Language Models. *ACM Computing Surveys*, 56(3):1–37.

A Introduction

A.1 Infeasibility of Brute Force

The large number of possible combinations makes it infeasible to estimate the difficulty of every possible C-Test and to select the one that comes closest to the target difficulty. For instance, only considering the gap placement without varying the gap size results in $\binom{n}{m} = \frac{n!}{m!(n-m)!}$ C-Tests (for placing m gaps across n words). The average run time of the 200 multi-layer perceptrons (MLP) we evaluated during our hyperparameter tuning (Appendix C.2) is 1.06 ms for each gap and 22.36 ms for the whole C-Test (using an *Intel Core™ i5-8400 CPU* with 6 x 2.80GHz). Consequently, trying out all possible C-Test combinations for 20 out of 40 possible gaps would already require approximately 97.74 years.

A.2 Primer on Mixed-Integer Programming

Similar to integer linear programming (ILP), the goal of mixed-integer programming (MIP) is to identify an optimal solution for a given problem that is described by a mathematical model.[17] The model generally consists of three components (Nocedal and Wright, 1999, p. 2). First, *variables* that each need to be assigned some value to form a potential solution. Second, an *objective function* of the variables that needs to be minimized or maximized. Third, *constraints* that need to be satisfied; represented by (in)equalities and functions containing the variables. In contrast to the output generated by LLMs, the global solution of MIPs provides mathematical guarantees on the optimality of the solution and the preservation of the constraints. In other words, as long as the model is feasible (i.e., contains no contradicting constraints) the found solution can be proven to be optimal and to satisfy all constraints.

Methods. Commonly used methods for solving MIP problems with above mathematical guarantees are branch-and-bound (Land and Doig, 1960), cutting planes (Gomory, 1960), and their combination (branch-and-cut). These algorithms consider the whole solution space of the problem but in a much more efficient manner than brute force. They furthermore ensure the optimality of the solution by only taking feasible solutions into account. To reduce the solution space efficiently, they only search for solutions that are better than the already found ones. This is done by solving a relaxation at each node that provides an estimate whether in the subproblem corresponding to the node, a better solution can be found. If not, the node is cut off; effectively reducing the solution space. Although these methods are usually much more efficient than brute force, one substantial limitation is the worst-case complexity.

Complexity. A key challenge of MIP is that ensuring the mathematical guarantees also results in NP-hard problems. As such, the general worst-case complexity has been shown to be exponential (see, e.g., Dash 2002) except for some cases with a proven polynomial worst-case complexity (see, e.g., Basu et al. 2023). Despite the high theoretical worst-case complexity, various methods have been developed to ensure that practical instances are solved in reasonable time. One such method is presolving (Achterberg et al., 2020). Presolving transforms the problem into one that is easier to solve, but has the same properties (theoretically) regarding feasibility and boundness; and furthermore, shares the same optimal solution. Presolving can substantially reduce the problem complexity, making MIP practically feasible. To show the practical feasibility of our approach, we provide statistics on the problem size and the impact of presolving, branch-and-bound, and cutting planes in Appendix D.1.3. We further discuss the importance of the actual problem formulation and devise alternate formulations to improve the run time of `MIP` in Appendix D.5.

A.3 `GPT-4`: Prompt and Response (Fig. 1)

Fig. 3 shows the prompt and response we used to generate a C-Test out of a short sentence. We find that GPT-4 fails to follow all constraints for C-Test generation; more specifically, that only parts of a word should be turned into a gap. Interestingly, the model provides a good explanation of the construction principle of C-Tests but cannot follow it.

B Task Instantiation

B.1 Feature description

Table 6 shows the full list of the features we used to train our gap error rate prediction model f_θ (cf. Beinborn 2016, pp. 219–224 for a full description of the features). Note, that the feature *PositionOfGap* is described as "[t]he position of the gap (indicated as the number of tokens)" (Beinborn 2016,

[17] This section provides a brief introduction into MIP. For more details, we refer the interested reader to Schrijver (1986) and Wolsey and Nemhauser (1998).

Generate a C-Test with difficulty 0.1 and 3 gaps using following sentence:
The cat sat on the bench today.

A C-Test is a type of language proficiency test where words in a given text are partially deleted, and test-takers are asked to complete the missing parts based on the context. The difficulty is often set by the number and choice of gaps.
For a difficulty of 0.1 with 3 gaps from the sentence "The cat sat on the bench today.", the C-test can be constructed as follows:
The c__ sat __ the b__ today.
The solutions are:
c__ = cat
__ = on
b__ = bench
Depending on the target audience, the difficulty might vary, but this provides a basic structure for a C-Test from the given sentence.

Figure 3: Prompt (top) and response (bottom) of GPT-4 (OpenAI, 2023) for the request to turn a short sentence into a C-Test. As can be seen, the word *on* is fully turned into a gap, showing that the model fails to follow all generation constraints for C-Tests.

page 224). In other words, this is the index of the respective token which is a constant that does not depend on the placement of the other gaps. Originally, *LengthOfSolutionInCharacters* describes the length of the word in characters that would remain constant (Beinborn 2016, page 220). To provide our model with a notion of the gap size, we change this feature to the length of the gap in characters. Both BERT-based features use the BERT$_{\text{base}}$ model (marked by ‡) and were proposed by Lee et al. (2020).

C Gap Difficulty Model

C.1 Transformer-based Models

We evaluate three different setups for the Transformer-based models (cf. Fig. 4).

- **MR** Our first setup trains the model in a masked-regression (MR) manner. For each gap in the C-Test, we insert the `[mask]` token. We then process the whole sentence as is; i.e., we tokenize it and feed resulting sequence into the model which is then trained to predict the gap error rate for each masked token. We further use a special label (-100) with a modified loss function to ensure that only the gaps are considered during training. An example is shown in Fig. 4a.
- **CLS** Our second setup only uses the `[cls]` token for gap error rate prediction. To do so, we turn each of the gap into a sentence where only the gap itself is masked and use the `[cls]` in a sentence regression manner (cf. Fig. 4b).
- **CLS+F** Finally, we additionally enrich the CLS setting with the 59 features proposed by Beinborn et al. (2014). For each gap we then concatenate the `[cls]` token with the feature

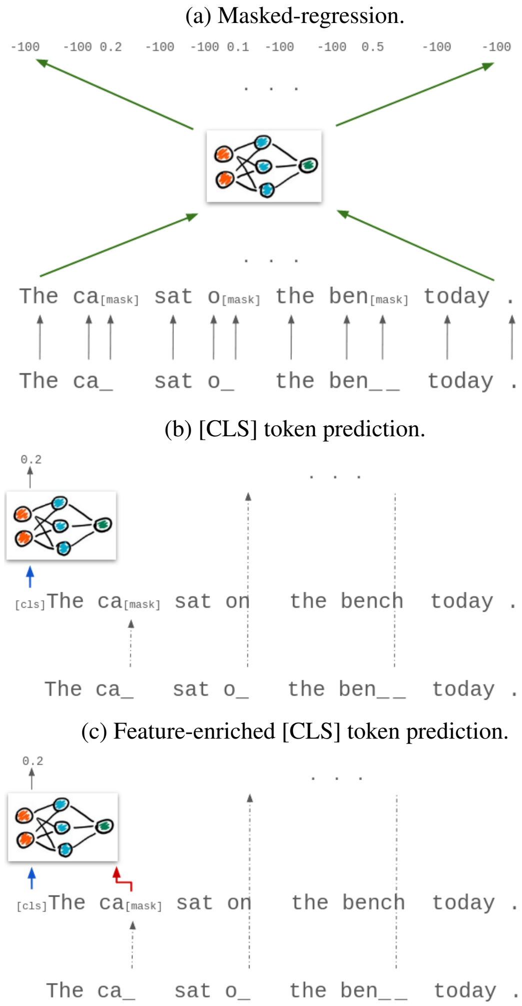


Figure 4: Transformer-based model setups.

Index k	Feature	Size	Placement	Type
0	AvgSentenceLength	-	-	Float
1	AvgWordLengthInCharacters	-	-	Float
2	AvgWordLengthInSyllables	-	-	Float
3	BigramSolutionRank	-	-	Integer
4	COPCognate_Exists	-	-	Binary
5	GapIsADJ	-	-	Binary
6	GapIsADV	-	-	Binary
7	GapIsART	-	-	Binary
8	GapIsCONJ	-	-	Binary
9	GapIsNN	-	-	Binary
10	GapIsNP	-	-	Binary
11	GapIsPP	-	-	Binary
12	GapIsPR	-	-	Binary
13	GapIsV	-	-	Binary
14	IsAcademicWord	-	-	Binary
15	IsCompound	-	-	Binary
16	IsDerivedAdjective	-	-	Binary
17	IsFunctionWord	-	-	Binary
18	IsInflectedAdjective	-	-	Binary
19	IsInflectedNoun	-	-	Binary
20	IsInflectedVerb	-	-	Binary
21	IsLemma	-	-	Binary
22	IsWordWithLatinRoot	-	-	Binary
23	LanguageModelProbability	-	-	Float
24	LanguageModelProbabilityOfPrefix	-	-	Float
25	LanguageModelProbabilityOfSolution	-	-	Float
26	LeftBigramLogProbability	-	-	Float
27	LeftTrigramLogProbability	-	-	Float
28	LmRankOfSolution	-	-	Float
29	MaxStringSimWithCandidate	-	-	Float
30	NrOfBigramCandidates	-	-	Integer
31	NrOfCandidates	-	-	Integer
32	NrOfTrigramCandidates	-	-	Integer
33	NrOfUbySenses	-	-	Integer
34	NrOfUnigramCandidates	-	-	Integer
35	NumberOfChunksPerSentence	-	-	Float
36	OccursAsText	-	-	Binary
37	PhoneticScore	-	-	Float
38	PhoneticSimilarity	-	-	Float
39	RightBigramLogProbability	-	-	Float
40	RightTrigramLogProbability	-	-	Float
41	TrigramLogProbability	-	-	Float
42	TrigramSolutionRank	-	-	Integer
43	TypeTokenRatio	-	-	Float
44	Uby_XDiceScore	-	-	Float
45	UnigramLogProbability	-	-	Float
46	UnigramSolutionRank	-	-	Float
47	VerbVariation	-	-	Float
48	posProbability	-	-	Float
49	‡BERT$_{\texttt{base-cased}}$ word prediction probability	✘	-	Float
50	‡BERT$_{\texttt{base-cased}}$ entropy(softmax(top50))	✘	-	Float
51	NumberOfGapsInCoverSentence	-	✘	Integer
52	NumberOfPrecedingGaps	-	✘	Integer
53	NumberOfPrecedingGapsInCoverSentence	-	✘	Integer
54	OccursAsGap	-	✘	Binary
55	PositionOfGap	-	-	Integer
56	IsCompoundBreak	✘	-	Binary
57	IsReferentialGap	✘	-	Binary
58	IsSyllableBreak	✘	-	Binary
59	LengthOfSolutionInCharacters	✘	-	Integer
60	LengthOfSolutionInSyllables	-	-	Integer

Table 6: Features of our model f_θ. ✘ marks a dependency of the feature on the size or placement of the gap. For clarity, we use the same nomenclature as Beinborn (2016).

vector to predict the gap error rate (cf. Fig. 4c). Note, that we exclude both BERT-based features in this setup, assuming that the models already have sufficient knowledge about the respective prediction probabilities of the masked gap.

C.2 Hyperparameter Tuning

We mainly tune hyperparameters for the SVM and MLP. For the SVM, we evaluate $c \in \{0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 10000, 100000\}$ and find that $c = 0.01$ performs best. For the MLP, we randomly generate 100 configurations where we sample $\{1, 2\}$ layers with $\{10, \ldots, 100\}$ hidden units and run each configuration twice, once with Linear and once with ReLU activation functions; using a batch size of 10. All neural models are further trained for 250 epochs with an initial learning-rate of 5e-5 using the AdamW optimizer (Loshchilov and Hutter, 2019) and the mean squared error (MSE) loss. The transformer-based models are trained with a batch size of 5. We use the development set (cf. Table 1) to select our model for evaluation on the test set; i.e, we use the model from the best performing epoch. The best performing MLP configuration for the Linear activation function has two layers with 87 hidden units in the first, and 91 in the second layer. The best performing MLP configuration for the ReLU activation function has 31 hidden units in the first and 16 in the second layer (with a total of two layers). We do not tune any hyperparameters for the XGB model, using the default configuration. We use Scikit-learn (LR and SVM, Pedregosa et al. 2011), the XGBoost library (Chen and Guestrin, 2016), and PyTorch (Paszke et al., 2019) with the transformers library (Wolf et al., 2020) to implement and train our models.

C.3 Results

Table 7 shows the full results of our gap difficulty model experiments on the development and test data. Overall, we find that XGB performs best on the test set in terms of RMSE and Pearson's ρ, and has the highest Pearson's ρ on the development set. Whereas $\text{MLP}_{\text{Linear}}$ has a lower RMSE on the development set, we find that the results substantially worsen on the test set, indicating that the models have overfitted on the training data. Interestingly, the feature-enriched Transformer-models seem to suffer from the same overfitting as the standard feature-based models, as they perform substantially worse on the test set. Finally, we find that the MR setup performs surprisingly well, showing the overall best performance across all Transformer-based setups, even outperforming the simpler CLS token setup. This indicates that considering the gap interdependencies (as done in the MR setup but not in the CLS setup) provides crucial knowledge to the model and should be investigated in more detail in future work.

C.3.1 Error analysis of $\text{DeBERTa}_{\text{v3-large}}$

Fig. 5 shows box plots on the differences between the predicted and true error rates (absolute values). We can see that $\text{DeBERTa}_{\text{v3-large}}$ performs better than XGB for the gap sizes one and three, but worse for gap sizes equal or larger than four. Plotting all predicted error rates (Fig. 6) shows that XGB covers a larger range with minimum and maximum values at 0.07 and 0.74 in contrast to $\text{DeBERTa}_{\text{v3-large}}$ with minimum and maximum values at 0.07 and 0.61. We further find that $\text{DeBERTa}_{\text{v3-large}}$ tends to predict too low error rates for 61.56% of the gaps whereas for XGB, this happens in only 57.81% of the cases, indicating that $\text{DeBERTa}_{\text{v3-large}}$ tends to underestimate the difficulty of the gaps.

C.3.2 Feature Ablation for XGB

While the features proposed by Beinborn (2016) are linguistically and pedagogically motivated, they have not been evaluated using XGB as the model. We thus conduct an ablation study to identify any features that may be removed without affecting the model's performance. For ablation, we follow the taxonomy of Beinborn (2016), who propose four feature categories:

Readability Captures the difficulty of the overall text. They are further categorized into surface-level (k=0, 1, 2), lexical-semantic (k=43, 47), and syntactic (k=35) features.

Word Difficulty Captures the individual difficulty of a word which is comprised of familiarity (k=33, 45, 59, 60), morphology & compound (k=15, 16, 18–21, 56, 58), syntax & context (k=5–13, 17, 26, 39, 41, 48), L1 influence (k=4, 14, 22, 44), and spelling difficulty (k=23–35, 36–38) features.

Candidate Ambiguity Considers ambiguity introduced by possible solution candidates and is divided into mirco-level (i.e., close-range; k=3, 29–32, 34, 42, 46) and marco-level (i.e.,

	Dev		Test	
Model	↓ RMSE	↑ Pearson's ρ	↓ RMSE	↑ Pearson's ρ
SVM (Beinborn et al., 2014)	**0.23**	0.50	—	—
SVM (Lee et al., 2019)	0.24	0.49	—	—
MLP (Lee et al., 2019)	0.25	0.42	—	—
BiLSTM (Lee et al., 2019)	0.24	0.49	—	—
$\text{SVM}_{\text{Linear},c=0.01}$	0.270	0.485	1.846	−0.111
MLP_{ReLU}	0.242	0.518	1.113	0.099
$\text{MLP}_{\text{Linear}}$	**0.232**	0.548	1.681	0.085
LR	0.239	0.559	2.949	−0.385
XGB	0.237	**0.614**	**0.285**	**0.439**
$\text{BERT}^{\text{MR}}_{\text{base}}$	0.249	0.420	0.311	0.279
$\text{BERT}^{\text{MR}}_{\text{large}}$	0.255	0.362	0.319	0.174
$\text{RoBERTa}^{\text{MR}}_{\text{base}}$	0.258	0.388	0.324	0.159
$\text{RoBERTa}^{\text{MR}}_{\text{large}}$	0.262	0.281	0.324	0.050
$\text{DeBERTa}^{\text{MR}}_{\text{v3-base}}$	0.248	0.502	0.311	0.245
$\text{DeBERTa}^{\text{MR}}_{\text{v3-large}}$	0.245	0.484	0.308	0.259
$\text{BERT}^{\text{CLS}}_{\text{large}}$	0.263	0.270	0.708	0.006
$\text{BERT}^{\text{CLS}}_{\text{base}}$	0.259	0.348	0.598	0.006
$\text{RoBERTa}^{\text{CLS}}_{\text{base}}$	0.268	0.287	0.456	-0.031
$\text{RoBERTa}^{\text{CLS}}_{\text{large}}$	0.262	0.284	0.573	0.014
$\text{DeBERTa}^{\text{CLS}}_{\text{v3-base}}$	0.268	0.283	0.469	-0.022
$\text{DeBERTa}^{\text{CLS}}_{\text{v3-large}}$	0.267	0.277	0.491	0.016
$\text{BERT}_{\text{CLS+F}_{\text{large}}}$	0.239	0.592	85.055	-0.001
$\text{BERT}^{\text{CLS+F}}_{\text{base}}$	0.245	0.570	60.432	-0.030
$\text{RoBERTa}^{\text{CLS+F}}_{\text{base}}$	0.239	0.587	100.689	-0.004
$\text{RoBERTa}^{\text{CLS+F}}_{\text{large}}$	0.241	0.592	82.148	0.022
$\text{DeBERTa}^{\text{CLS+F}}_{\text{v3-base}}$	0.243	0.579	72.880	0.012
$\text{DeBERTa}^{\text{CLS+F}}_{\text{v3-large}}$	0.241	0.585	64.031	-0.015

Table 7: Root mean squared error (RMSE) and Pearson's correlation ρ for predicting the gap error rate across different models. The first four rows show the results reported in the respective work (hence, there are no results on the test portion of our data); all other rows report the results of our experiments. All results are averaged over ten runs with different random seeds. Overall, we find that XGB performs best on the test data.

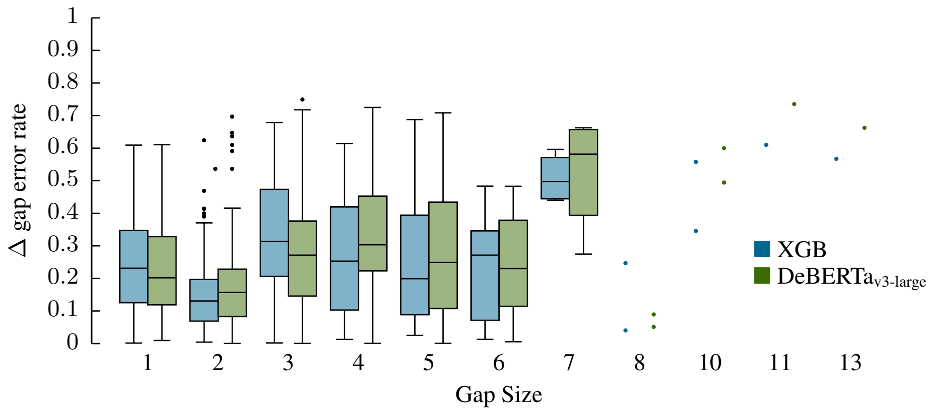


Figure 5: Absolute differences between the predicted to the true error rate (Δ gap error rate) sorted by gap size.

long-range; k=28) features.

Item Dependency Captures dependencies between different items by using position (k=51–53, 55), neighbor (k=27, 40, 54), and referentiality (k=57) features.

In addition, we conduct an extra ablation experiment for the BERT-based (k=49, 50) features proposed by Lee et al. (2020). Table 8 shows the impact of removing a whole feature category or individual feature subsets on the XGB model. Interestingly, we can observe multiple instances where

	Dev		Test	
Feature	↓ RMSE	↑ Pearson's ρ	↓ RMSE	↑ Pearson's ρ
All	0.240	0.533	0.285	0.439
– Readability	**+0.023**	**+0.090**	-0.051	-0.353
– Surface	-0.006	-0.003	-0.044	-0.290
– Lexical-semantic	-0.019	-0.052	-0.047	-0.480
– Syntactic	-0.001	-0.006	-0.018	-0.089
– Word Difficulty	-0.020	-0.154	-0.068	-0.285
– Familiarity	-0.020	-0.106	-0.043	- 0.386
– Morphology & Compounds	**+0.003**	**+0.056**	-0.019	-0.089
– Syntax & Context	-0.013	**+0.013**	-0.030	-0.241
– L1 Influence	-0.010	-0.100	-0.036	-0.307
– Spelling Difficulty	-0.029	-0.123	-0.034	-0.265
– Candidate Ambiguity	-0.011	-0.002	-0.092	-0.279
– Micro-level	-0.034	-0.110	-0.044	-0.471
– Macro-level	-0.010	-0.027	-0.104	-0.230
– Item Dependency	- 0.005	**+0.006**	-0.053	-0.465
– Position	**+0.005**	**+0.022**	-0.038	-0.350
– Neighbor Effects	-0.004	**+0.019**	-0.052	-0.406
– Referentiality	**+0.006**	**+0.020**	-0.038	-0.376
– BERT	-0.015	-0.096	-0.052	-0.349

Table 8: Ablation study for different subsets of features using the XGB model. Interestingly, removing some of the features improves the model's performance on the development data but substantially deteriorates the performance on the test data.

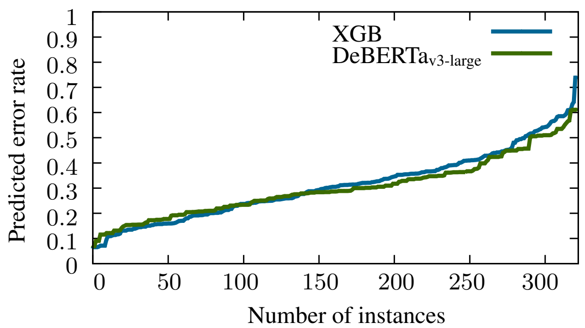


Figure 6: Predicted gap error rates of XGB and DeBERTa$_{v3\text{-large}}$. We find that XGB covers a larger range of predicted error rates than DeBERTa$_{v3\text{-large}}$, especially with respect to the maximum predicted values (0.74 vs 0.61).

removing a set of features improves the model's performance on the development data; especially the removal of all readability features that improves the rooted mean squared error (RMSE) by 0.023 and Pearson's ρ by 0.09. Moreover, their impact on the performance seems to be interdependent, as their individual removal decreases both scores. We can also see that the removal of item dependency features (and subsets) consistently improves Pearson's ρ. In contrast, we observe substantial drops especially in terms of Pearson's ρ on the test set which deviates from the commonly used static C-Test format present in the training and development sets. We thus conclude that the features proposed by Beinborn (2016) and Lee et al. (2020) substantially assist the model in generalizing to different gap sizes and placements.

D User Study

D.1 Setup

This section provides further details with respect to the setup of our user study.

D.1.1 Reimplementing `SEL` and `SIZE`

Given the high performance of the XGB model, reimplementing the `SEL` and `SIZE` strategies with the XGB model may result in better C-Tests. We thus reimplement both strategies using the XGB model and estimate the performance between different generation strategies by measuring the variability in terms of edit distance of the resulting C-Test (i.e., the total number of characters turned into a gap).

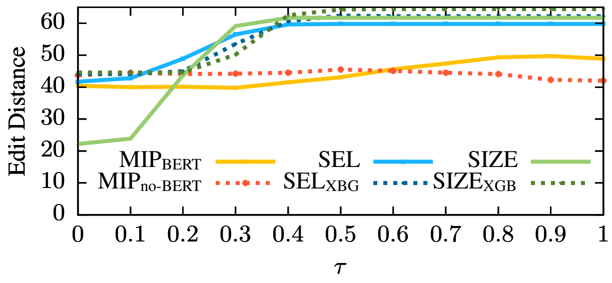


Figure 7: Edit distance (total gap size per C-Test); averaged across 100 C-Tests for different target difficulties.

Original baselines. Lee et al. (2019) define `SEL` and `SIZE` as follows:

- **`SEL`** estimates the difficulty of all $n = 40$ candidate gaps using an SVM trained on the 59 hand-crafted features defined by Beinborn (2016). All gaps are then divided into two sets; each set only consists of gaps that are easier (or harder) than the target difficulty τ. Each set is then sorted according to the difference (i.e., the distance) between a gap's difficulty and τ. Finally, the C-Test is generated by selecting the gaps closest to τ from each set in an alternating manner, until the final number of gaps $m = 20$ have been selected. In the reimplementation, we replace the original model with our XGB model.
- **`SIZE`** estimates the gap difficulty of all default gaps (using the same model as in SEL). In addition, Lee et al. (2019) train two SVMs that predict the relative change in difficulty if the gap was increased (decreased) by one character. To reduce the time required for feature processing, both models only use six features that least degrade the performance (identified in an ablation study). Using these models, the gap size is then either increased (decreased) until the target difficulty τ is achieved. For the reimplementation, we replace the relative prediction model with our XGB model and directly predict the difficulty of a gap that is increased (or decreased) by one character. This is made possible due to some improvements (in terms of run time efficiency) we added to the feature pipeline provided by Beinborn (2016).[18] As in the original algorithm, we do this until we achieve a gap difficulty of larger (smaller) than τ.

[18] We obtained permission to share an executable `.jar` file of the improved pipeline under an open source license.

Experimental setup. Note, that we cannot use our test data as it has been generated by the very same generation strategies (`SEL` and `SIZE`) we aim to evaluate and would distort the results. For the experiments, we randomly sample 100 passages from the GUM corpus (Zeldes, 2017). As this corpus was designed as an educational corpus for pedagogical use, we conjecture that it better fits our use case than other open corpora comprised of news articles or Wikipedia articles. We further investigate the helpfulness of varying the BERT features ($k = 49$ and $k = 50$) as recomputing them leads to the highest overhead in terms of computation time (see Appendix D.1.2).

Results. Fig. 7 shows the results of our approach with ($\texttt{MIP}_{\text{BERT}}$) and without ($\texttt{MIP}_{\text{no-BERT}}$) varying the BERT features, `SEL` and `SIZE` as proposed by Lee et al. (2019), and our reimplementation using the XGB model ($\texttt{SEL}_{\text{XGB}}$ and $\texttt{SIZE}_{\text{XGB}}$). Overall, we find that varying the BERT features leads to a higher variability with smaller edit distances for lower target difficulties, and larger edit distances for higher target difficulties. We further find that `SEL` results in substantially smaller edit distances for lower difficulties than $\texttt{SIZE}_{\text{XGB}}$; indicating that the relative difficulty model proposed by Lee et al. (2019) may lead to better C-Tests especially for easy ones. We do not see major differences between `SEL` and $\texttt{SEL}_{\text{XGB}}$, but find that `SEL` results in slightly lower minimum edit distances. For the study, we thus use the `SEL` and `SIZE` implementations without any changes, as they display a higher variability, and hence, are stronger baselines.

D.1.2 Run time of MIP, SEL, and SIZE

While the optimality of the solution found by `MIP` is ensured by algorithm design (cf. Appendix A.2), one frequent limitation of constraint optimization approaches are their potentially long run times until finding a feasible solution. To show the feasibility of our approach, we measure the run time of the gap variability experiments conducted in the previous section (Appendix D.1.1). All experiments were conducted using an *Intel Core™ i5-8400 CPU* with 6 x 2.80GHz.

Results. For MIP_{BERT}, the solver requires 22.5 seconds on average to find an optimal solution. We further find that not varying the BERT features can lead to a substantial speedup by 16.9 seconds; reducing the run time of the solver to 5.6 seconds. In addition, computing the varying BERT features without a GPU leads to an additional overhead of 25.5 seconds. In total, not varying the BERT features thus reduces the run time by 42.4 seconds due to the smaller number of decision variables ($2 \cdot 20$) and static BERT features. Nonetheless, we find that the 48.6 seconds of MIP_{BERT} against the 6.2 seconds of $\text{MIP}_{\text{no-BERT}}$ still remains within an acceptable range; especially when considering the better results of MIP_{BERT}. The run times of `SEL` and `SIZE` are on average 14.6 seconds and 15.3 seconds, respectively; resulting in a difference of ~ 30 seconds between `MIP` and the baselines.[19]

Discussion. Although the difference of ~ 30 seconds may seem substantial at first, `MIP` has two inherent advantages over `SIZE` (`SEL`). First, the theoretical guarantees that come with the mixed-integer formulation ensure that under a given model f_θ, it is guaranteed that the resulting C-Test is optimal. Same cannot be considered for all other baselines, as the solution space is reduced approximatively for `SIZE` (`SEL`). Moreover, selecting or excluding specific words as well as limiting the gap size can be done by simply adding constraints while the very same guarantees are being kept (i.e., the resulting solution is optimal for model f_θ). For instance, turning a word at position i into a gap of size j only requires adding the constraint $s_{i,j} = 1$. Second, `MIP` considers the whole solution space during generation. Consequently, this allows us to add constraints across the whole solution space. In contrast, changing the gap placement in `SIZE` (gap size in `SEL`) would require a manual assessment and adaptation of the C-Test as the placement (gap size) is not considered during generation. Besides the improvements in terms of optimization objective we propose in Appendix D.5, the run time—especially for the feature generation—could be further improved by incorporating more efficient models such as TinyBERT (Jiao et al., 2020) in addition with other methods that improve inference efficiency (Treviso et al., 2023). This remains to be investigated in future work.

[19] While we cannot provide run time measurements for `GPT-4` due to too many instabilities regarding the latency, we provide some estimates based on Llama-2 (Touvron et al., 2023). We find that our prompts would consist of 1277.75 input tokens (on average). Using the model and the inference speed reported by Touvron et al. (2023, p. 48), then results in an approximate run time of 1277.75 tokens $\cdot 25 \frac{\text{ms}}{\text{token}} \approx 31.9$ seconds on eight NVIDIA A100 GPUs with 80 GB RAM.

	μ	$\pm$	σ
Model Statistics			
Rows	4,413.95	$\pm$	4.02
Columns	179,491.45	$\pm$	24.06
Nonzeros	179,486.50	$\pm$	566.61
Constraints	21,887.62	$\pm$	1,484.86
Variables (before presolve)			
Continuous	6,563.00	$\pm$	0.00
Integer	172,928.45	$\pm$	24.06
Binary	172,928.45	$\pm$	24.06
Presolved			
Rows	8,854.53	$\pm$	749.26
Columns	6,852.54	$\pm$	550.05
Nonzeros	46,695.43	$\pm$	5,788.68
Variables (after presolve)			
Continuous	1,248.73	$\pm$	101.67
Integer	5,603.81	$\pm$	459.85
Binary	5,496.23	$\pm$	455.90
Cutting Planes	4,823.34	$\pm$	3,219.46
B & B Nodes	2,336.07	$\pm$	5,916.47

Table 9: Average (μ) statistics about the optimization model, number of constraints, and variables. Presolving consistently and substantially reduces the number of variables. Most affected by the individual problem complexity are the the number of visited nodes and cutting planes with a high standard deviation (σ).

D.1.3 Empirical Complexity of MIP

As discussed in Appendix A.2, the employed methods do not affect the worst-case complexity and the problem itself remains NP-hard. However, analyzing the actual complexity of the experiments conducted in Appendix D.1.1 reveals that practical instances are being solved in reasonable time. As can be seen in Table 9, the standard deviations with respect to the model statistics, the number of constraints, and variables are low; indicating a stable level of problem complexity. Moreover, we find that presolving consistently and substantially reduces the problem complexity. Finally, we can observe that the individual problem complexity primarily affects the final steps of solving the problem; i.e., the number of cutting plane cuts and visited branch-and-bound (B & B) nodes. In practice, all our experiments terminated and in feasible time (cf. Appendix D.5).

$\mathcal{T}$	Metric	μ	σ	C-Test	
$\mathcal{T}_1$	ARI	21.04	8.49	34.21	✗
	Coleman-Liau	12.65	1.87	12.84	✓
	DaleChallIndex	11.26	0.98	11.97	✓
	FleschReadingEase	38.58	23.02	10.37	✗
	GunningFogIndex	22.24	6.95	33.08	✗
	Kincaid	17.28	6.97	27.88	✗
	LIX	64.52	17.59	93.02	✗
	RIX	10.35	4.90	18.33	✗
	SMOGIndex	16.96	3.69	21.71	✗
$\mathcal{T}_2$	ARI	10.3	3.85	14.12	✓
	Coleman-Liau	9.43	2.21	10.63	✓
	DaleChallIndex	9.64	0.94	9.54	✓
	FleschReadingEase	69.26	15.77	57.86	✓
	GunningFogIndex	12.87	3.32	16.43	✗
	Kincaid	8.69	3.39	11.90	✓
	LIX	41.93	8.91	51.44	✗
	RIX	4.42	1.90	6.60	✗
	SMOGIndex	11.27	2.19	13.68	✗
$\mathcal{T}_3$	ARI	14.96	6.14	21.07	✓
	Coleman-Liau	10.33	1.77	8.67	✓
	DaleChallIndex	10.12	0.80	9.30	✗
	FleschReadingEase	57.85	16.36	54.60	✓
	GunningFogIndex	17.16	4.99	21.70	✓
	Kincaid	12.4	4.96	16.76	✓
	LIX	51.52	12.31	62.63	✓
	RIX	6.52	2.84	8.00	✓
	SMOGIndex	13.71	2.67	14.29	✓
$\mathcal{T}_4$	ARI	14.98	5.19	15.92	✓
	Coleman-Liau	11.14	1.51	10.45	✓
	DaleChallIndex	10.24	0.77	9.87	✓
	FleschReadingEase	53.85	14.27	55.65	✓
	GunningFogIndex	17.33	4.27	17.91	✓
	Kincaid	12.65	4.22	13.23	✓
	LIX	52.52	10.99	55.22	✓
	RIX	6.94	2.83	7.50	✓
	SMOGIndex	14.19	2.37	14.29	✓

Table 10: Readability scores of $\mathcal{T}_1$–$\mathcal{T}_4$ on eight different metrics.

D.1.4 Text Selection

As we expect most of our participants to have at least a college degree, we focus on books that presumable have a higher reading difficulty. The selected books are (in alphabetical order):

- $\mathcal{T}_1$ Beyond Good and Evil (Friedrich Nietzsche)
- $\mathcal{T}_2$ Crime and Punishment (Fyodor Dostoevsky)
- $\mathcal{T}_3$ Emma (Jane Austen)
- $\mathcal{T}_4$ Pride and Prejudice (Jane Austen)

To evaluate how well the randomly sampled passages in the study represent each book, we conduct an analysis with respect to their reading difficulty and compare each passage against all possible passages that follow our selection criteria (cf. **Text selection** in Section 6). Table 10 shows the scores for eight reading difficulty metrics that were computed using a respective python package.[20] For each text ($\mathcal{T}$), we show the average (μ) and standard deviation (σ) for the reading difficulty of all paragraphs and the one used in the study (C-Test). The last column indicates if the paragraph used in the study falls within (✓) or outside (✗) the boundary of $\mu \pm \sigma$. Overall, we see that all selected text passages have a high reading difficulty satisfying our criteria of college graduate level or higher reading difficulty. Moreover, we see that the text passages (except $\mathcal{T}_1$) fall within the boundary of average and standard deviation across a majority of metrics, indicating that they represent the text well in terms of reading difficulty.

D.2 Design

D.2.1 Group Configurations

Table 11 shows the eight configurations used in our study where each participant solves C-Tests generated from each model $\mathcal{M}$, text $\mathcal{T}$, and two target difficulties τ.

	$\mathcal{T}_1$		$\mathcal{T}_2$		$\mathcal{T}_3$		$\mathcal{T}_4$	
	M	τ	M	τ	M	τ	M	τ
1	M_1	0.1	M_4	0.9	M_2	0.1	M_3	0.9
2	M_1	0.9	M_2	0.1	M_4	0.9	M_3	0.1
3	M_2	0.1	M_1	0.9	M_3	0.9	M_4	0.1
4	M_2	0.9	M_1	0.1	M_3	0.1	M_4	0.9
5	M_3	0.1	M_2	0.9	M_4	0.1	M_1	0.9
6	M_3	0.9	M_4	0.1	M_1	0.9	M_2	0.1
7	M_4	0.1	M_3	0.9	M_2	0.9	M_1	0.1
8	M_4	0.9	M_3	0.1	M_1	0.1	M_2	0.9

Table 11: Configurations used in our user study. As can be seen, all generation strategies M and target difficulties τ are evenly distributed. Our models are: `GPT-4` (M_1), `MIP` (M_2), `SEL` (M_3), and `SIZE` (M_4).

D.2.2 Study Examples and Questionnaire

Fig. 8 shows all questions participants were asked to answer upon registration. An example C-Test is provided in Fig. 9. Fig. 10 shows the feedback our participants were asked to provide after each C-Test.

D.3 Results

This section provides a detailed analysis about our study participants and their responses in the questionnaire, as well as details about the GAMM formulation and the resulting p-values.

[20] https://pypi.org/project/readability/

Q1: Please estimate your current language proficiency in English
A1: ○ *Beginner (A1)* ○ *Elementary (A2)* ○ *Intermediate (B1)* ○ *Upper Intermediate (B2)* ○ *Advanced (C1)* ○ *Proficient (C2)*

Q2: I studied English for about ___ years.

Q3: How often do you practice or speak English?
A3: ○ *Never* ○ *Monthly* ○ *Weekly* ○ *Daily*

Q4: What is your native language?
A4: ____________________

Q5: Have you tried learning languages (other than English)? If yes, than which ones?
A5: ○ *Yes*, ________. ○ *No*.

Figure 8: Study questionnaire.

D.3.1 Participants

The study was distributed across our university and took place between the 21st of August 2023 and 8th of September 2023. All our participants are volunteers (and received no compensation) with an academic background with at least one college degree. They have achieved at least B1 proficiency in English based on the common European framework of references for languages (CEFR, Council of Europe 2001). Overall, two participants stated to have a B1 proficiency, nine a B2 proficiency, and 16 and 13 a C1 and C2 proficiency, respectively. On average, our participants have used English for 10 (± 6.5) years; mostly on a daily (27) or weekly (8) basis. Only two participants use English on a monthly basis, while three responded with never. Most of our participants are native German speakers (26). Other native languages were Chinese (3), Russian (2), Turkish (2), Arabic, Croatian, Italian, Hindi, Korean, Kyrgyz (together with Russian), Spanish, and Vietnamese (1 each). Finally, the majority (36) of our participants have attempted to learn a different language—on average 1.97 languages excluding English. Fig. 11 and Fig. 12 show the average scores and time taken for each participant, sorted by their provided CEFR self-estimate. Interestingly, we do not find any significant differences between the proficiency, score, and time taken, indicating that our participants actually had a similar English proficiency. This is in line with our observations from the GAMM analysis that indicates that none of our smooth terms play a significant role for modeling (cf. Appendix D.3.2).

D.3.2 GAMM Details for $\hat{\tau}$

We formulate our GAM model for the actual difficulty $\hat{\tau}$ as follows:

$$\begin{aligned}\hat{\tau} =& \mathrm{M} + \tau + f_1(\mathcal{T}) \cdot \mathrm{Z}_1 + f_2(\mathrm{CEFR}) \cdot \mathrm{Z}_2 \\ &+ f_3(\mathrm{User}) \cdot \mathrm{Z}_3 + f_4(\mathrm{Years}) \\ &+ \mathrm{M} \cdot \tau + \epsilon,\end{aligned} \tag{11}$$

where M represents the generation strategy and τ the target difficulty. $\mathcal{T}$, CEFR, User, and Years are terms for categorical values of text, language proficiency, participant, and number of years participants have been using English, modeled as smooth terms $f(\cdot)$ with a random effect Z.[21] Finally, $\mathrm{M} \cdot \tau$ models the interaction between generation strategy and target difficulty and ϵ is an unknown vector of random errors.

We use the `mgcv` (Wood et al., 2016) implementation available in R (R Core Team, 2022). Fig. 16 show the p-values between different generation strategies computed using the Wald test (Wald, 1943). Analyzing our GAMM shows that the model's explanatory power is substantial with $R^2 = 0.59$. Table 12 shows the F-test statistics (Bruning and Kintz, 1987) of the GAM model with respect to the parametric fixed terms. We find that all fixed terms are statistically significant. Besides M which has been discussed in the main paper, we further find that τ has a significant and positive effect on the response variable $\hat{\tau}$ with $\beta = 0.14, 95\%\mathrm{CI}[0.05, 0.23]$ and $p = 0.002$. Interestingly, we find that all modeled smooth terms are not significant as shown in Table 13. This however, is in line with our findings that indicate that our participants' CEFR self-estimate and their number of years using English do not substantially impact the score. Finally, the underlying text as well as the participant (User) do not substantially impact the score and thus, can be excluded as confounding factors in our observations.[22]

D.3.3 Perceived Difficulty

Fig. 19 and Fig. 20 show the box plots of our participant's responses for the perceived difficulty. We can see that `MIP` generated C-Tests that were perceived easiest by our participants. In addition, we again find a clear tendency that the easy C-Tests generated by `GPT-4` were also perceived as more

[21]Note, that we do not require a random effect for the years as they are quantifiable numeric values.

[22]Note, that this only applies to this specific study; in future studies, all these factors may impact the outcomes.

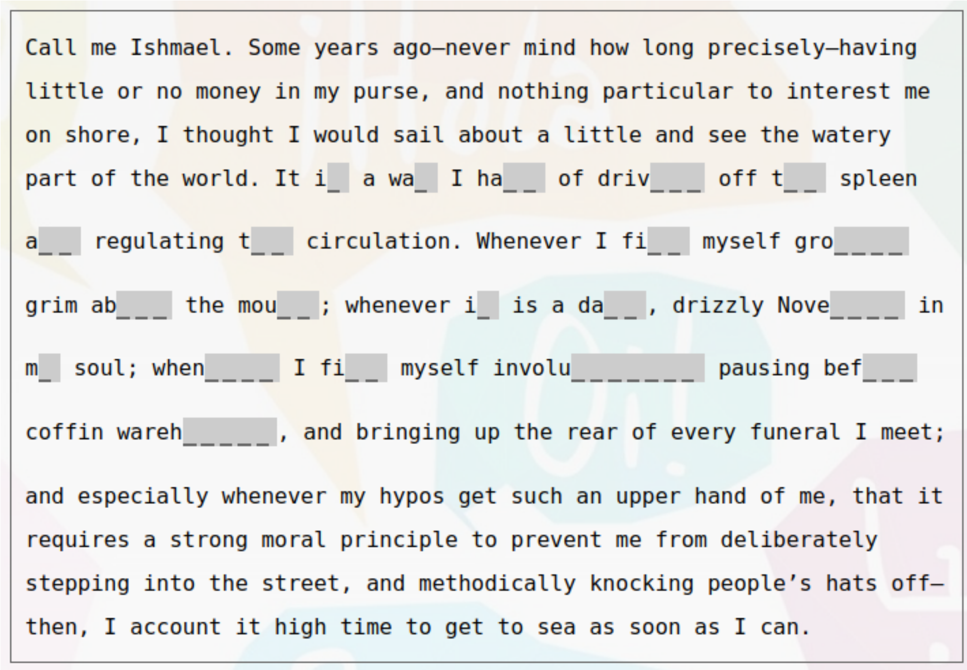


Figure 9: Example C-Test showing the interface used in our user study.

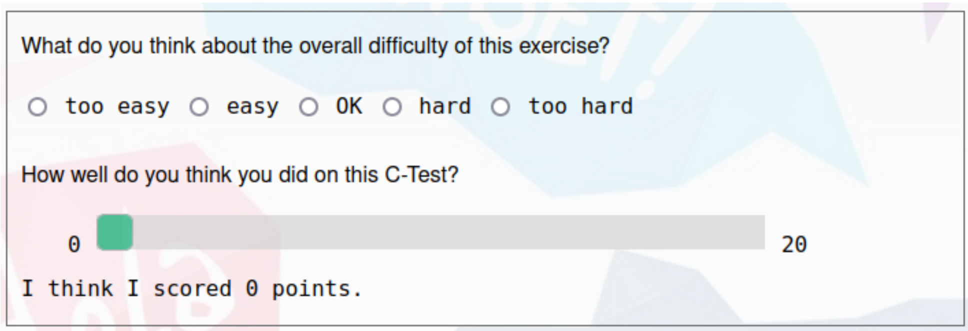


Figure 10: Collected feedback after each C-Test.

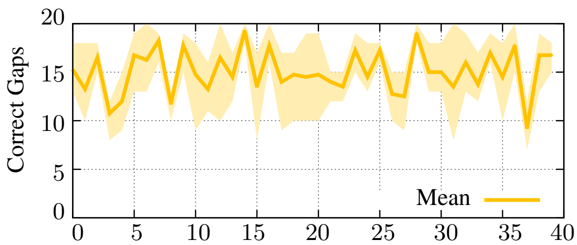


Figure 11: Average score for each participant sorted by their CEFR self-estimate. Shaded areas indicate the maximum and minimum scores.

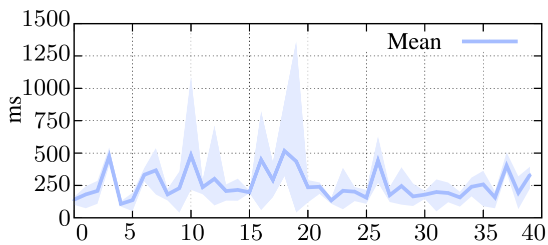


Figure 12: Average time taken for each participant sorted by their CEFR self-estimate. Shaded areas indicate the maximum and minimum time taken.

difficult than the hard C-Tests. Finally, conducting Wald tests (Wald, 1943) using our GAMM shows that `MIP` significantly outperform `SEL` and `GPT-4` and performs on-par with `SIZE` (cf. Fig. 17). Note, that `GPT-4` again performs significantly worse than all other generation strategies.

GAMM statistics. Table 14 and Table 15 show the analysis of our GAM model with respect to the perceived difficulty. We observe similar trends as

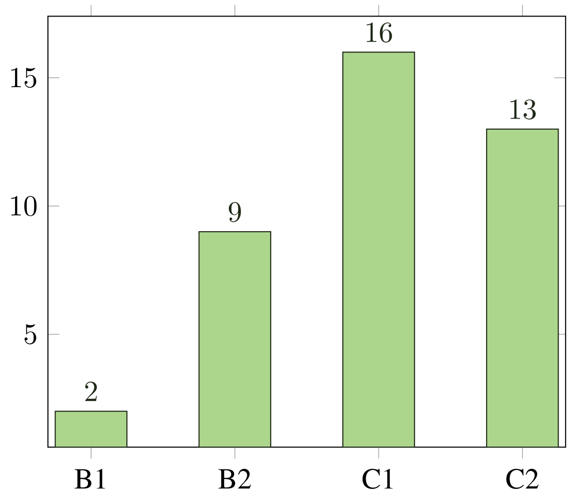


Figure 13: CEFR self-estimates of our participants.

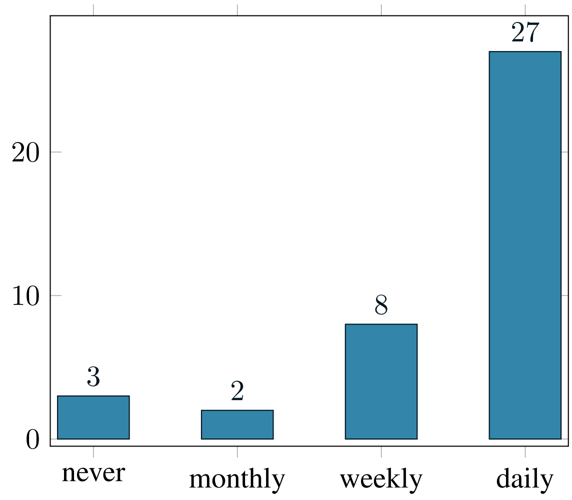


Figure 14: Frequency our participants use English.

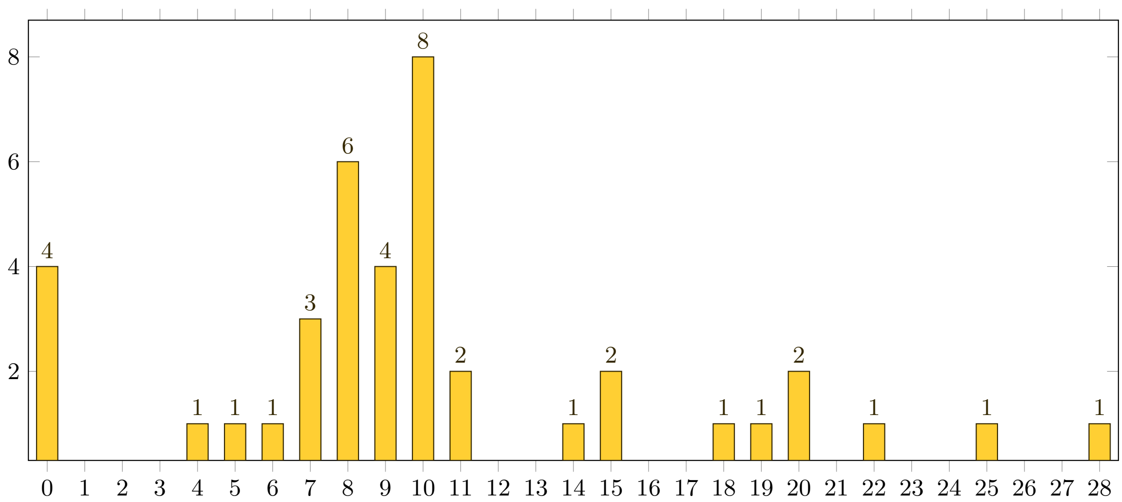


Figure 15: Number of years our participants have been using English.

$\hat{\tau}$	df	F	p-value
M	3.00	19.87	1.53e-10*
τ	1.00	9.57	0.00246*
$\mathrm{M} \cdot \tau$	3.00	9.96	6.47e-06*

Table 12: F-test statistics for the parametric fixed terms of the GAMM. df denotes the degrees of freedom. All terms are statistically significant.

$\hat{\tau}$	edf	Ref.df	F	p-value
$f_1(\mathcal{T})$	1.7013	4.00	0.983	0.990
f_2(CEFR)	0.8445	1.00	131.858	0.997
f_2(User)	27.3925	40.00	4.396	0.998
f_2(Years)	1.0000	1.00	0.013	0.908

Table 13: F-test statistics for the smooth terms of the GAM model. With edf denoting the effective degrees of freedom, and Ref.df the reference degrees of freedom. None of the terms are statistically significant, indicating that they do not have any impact on $\hat{\tau}$.

for $\hat{\tau}$ with an even higher substantial explanatory power $\mathrm{R}^2 = 0.81$.

D.4 Error Analysis

Details `MIP` vs `SIZE`. Table 16 show the number of occurrences of successive gaps (i.e., multiple gaps that occur in succession). Although it seems reasonable that successive gaps should be more difficult to fill out as less words provide context, we do not see substantial differences between C-Tests generated with $\tau = 0.1$ and $\tau = 0.9$. This indicates that the XGB model may not have a good notion about how successive gaps impact the overall C-Test difficulty.

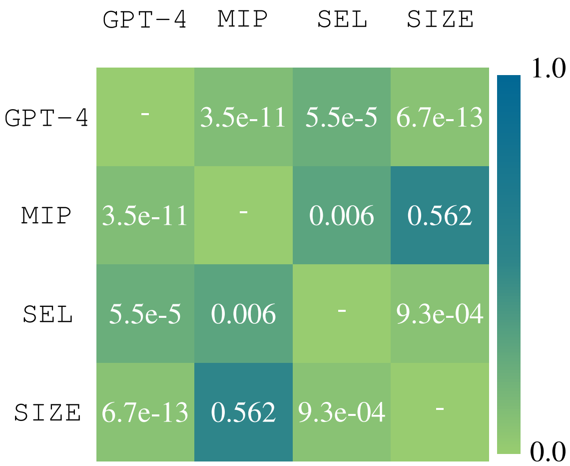


Figure 16: P-values of the Wald test between different C-Test generation strategies for $\hat{\tau}$. As can be seen, all strategies significantly outperform `GPT-4`, and `MIP` significantly outperforms `SEL`. We find no significant differences between `MIP` and `SIZE`.

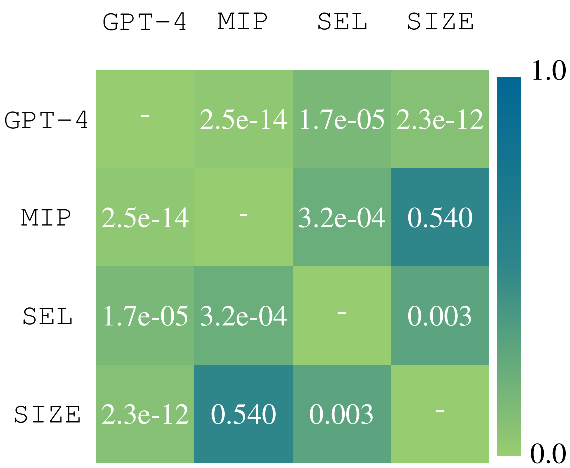


Figure 17: P-values of the Wald test between different C-Test generation strategies for the perceived difficulty. As can be seen, all strategies significantly outperform `GPT-4`, and `MIP` significantly outperforms `SEL`. We find no significant differences between `MIP` and `SIZE`.

Feedback	df	F	p-value
M	3.00	21.698	2.16e-11*
τ	1.00	23.871	3.07e-06*
M $\cdot$ τ	3.00	9.269	1.39e-05*

Table 14: F-test statistics for the parametric fixed terms of the GAMM. df denotes the degrees of freedom. All terms are statistically significant.

Feedback	edf	Ref.df	F	p-value
$f_1(\mathcal{T})$	2.825e+00	4.00	39.469	0.843
f_2(CEFR)	1.817e-06	4.00	0.000	1.000
f_2(User)	2.062e+01	40.00	1.313	0.991
f_2(Years)	2.771e+00	3.05	0.920	0.457

Table 15: F-test statistics for the smooth terms of the GAM model, with edf denoting the effective degrees of freedom, and Ref.df the reference degrees of freedom. None of the terms are statistically significant, indicating that they do not have any impact on the perceived difficulty.

`GPT-4` shortcomings. Fig. 21 shows the C-Tests generated by `GPT-4` for $\mathcal{T}_3$ with $\tau = 0.1$ and $\tau = 0.9$. As the lower gap error rates indicate, the more difficult C-Test is comprised of easier gaps, including five gaps of gap size one (compared to two for the easier C-Test). We can further observe that the gaps are primarily clustered around the beginning of the C-Test. In addition, we find that GPT-4 generates misleading explanation about its notion of difficulty, for instance:

Based on the provided examples and the desired difficulty, we'll try to generate a C-Test with 20 gaps and a difficulty of 0.1. This means that about 10% of the gaps should be moderately challenging to fill in.

Although the explanation is convincing, considering the fact that the model simply tries to add (remove) gaps to increase (decrease) the difficulty of the resulting C-Test makes it clear that the models does not have a notion of gap-level difficulty. However, the provided explanation may even mislead students to believe that they are actually solving a C-Test of a specific difficulty, resulting in a wrong self-assessment.

Gap size vs gap placement. Motivated by the substantial differences between `SEL` and `SIZE`, we conduct an analysis to see if their differences can be attributed to either one; gap size or gap placement. To better assess the differences between

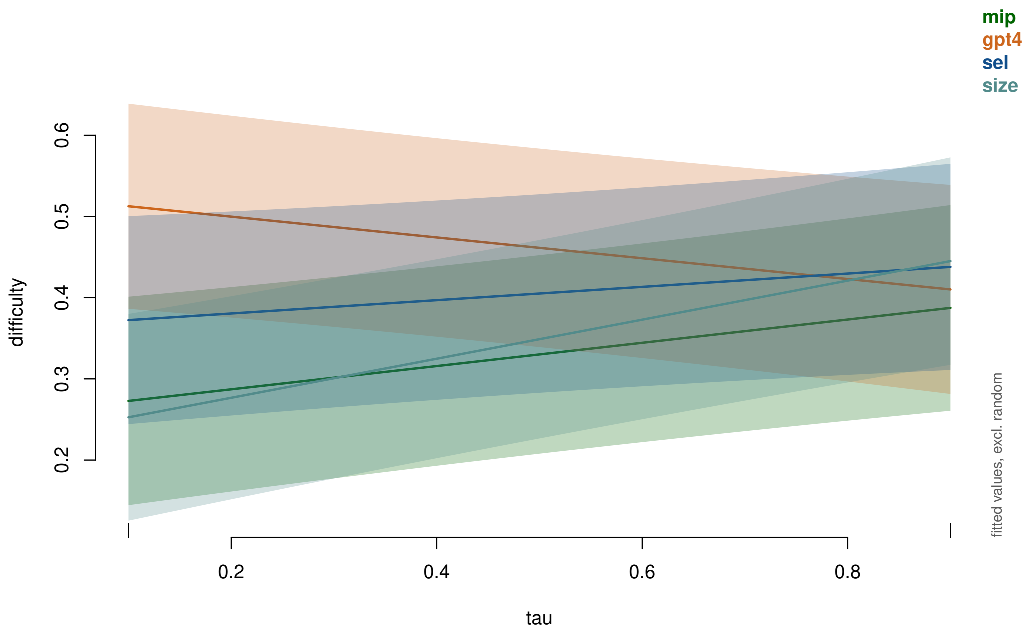


Figure 18: Linear regression curves for each generation strategy and difficulty. As can be seen, GPT-4 is the only strategy that shows a negative slope; indicating that the model as an inverted notion of difficulty.

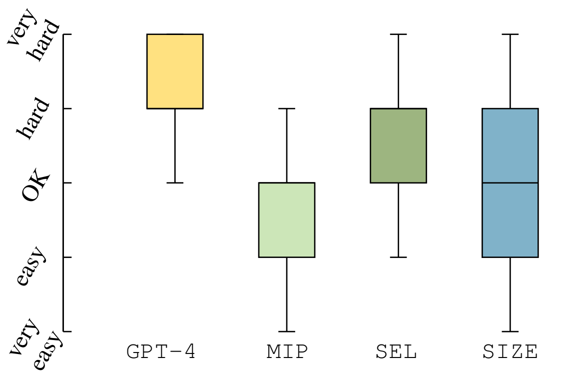


Figure 19: Perceived difficulty for $\tau = 0.1$ ($\sim$very easy).

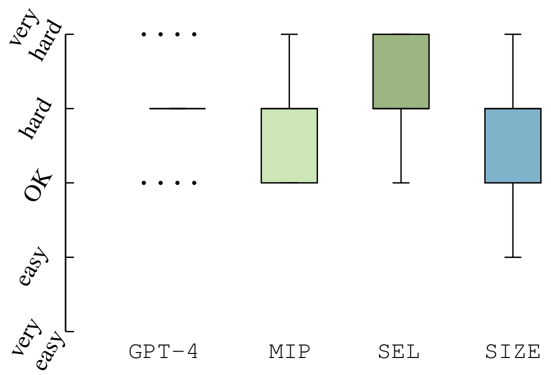


Figure 20: Perceived difficulty for $\tau = 0.9$ ($\sim$very hard).

$\tau = 0.1$

# Succ.	$\mathcal{T}_1$	$\mathcal{T}_2$	$\mathcal{T}_3$	$\mathcal{T}_4$
1	3	5	10	6
2	5	1	1	1
3	1	-	1	1
4	1	-	-	1
5	-	-	1	1
6	-	1	-	-
7	-	1	-	-
8	-	-	-	-
9	-	-	-	-
10	-	-	-	-

$\tau = 0.9$

# Succ.	$\mathcal{T}_1$	$\mathcal{T}_2$	$\mathcal{T}_3$	$\mathcal{T}_4$
1	4	6	7	3
2	4	2	2	2
3	-	2	-	1
4	2	1	1	-
5	-	-	1	-
6	-	-	-	-
7	-	-	-	-
8	-	-	-	-
9	-	-	-	-
10	-	-	-	1

Table 16: Number of occurrences for successive gaps (# Succ.) in C-Tests generated by MIP for $\tau = 0.1$ (left) and $\tau = 0.9$ (right).

```
...From t__ expense of the child,
how_____, he was so__ relieved.  The
bo_ had, wit_ the add________
softening cl____ of a lin_______ illness
of his mo_____'s, been the me___ of a
so__ of reconciliations; and Mr.  and
Mrs.  Ch_________, having no ch_______ of
their ow_, nor any ot___ young
cr_______ of equal ki_____ to care for,
offered to ta___ the whole ch____ of
the little Fr___ soon after...
```

(a) C-test of $\mathcal{T}_3$ generated with `GPT-4` for $\tau = 0.1$

```
...From t__ expense o_ the ch___,
how_____, he w__ soon rel______.  The
bo_ had, with t__ addit______
soften____ claim o_ a ling______
illness o_ his mo_____'s, be__ the
me___ of a so__ of recon__________;
and M_.  and M_.  Churchill, having
no children of their own, nor any other
young creature of equal kindred to care
for, offered to take the whole charge of
the little Frank soon after...
```

(b) C-test of $\mathcal{T}_3$ generated with `GPT-4` for $\tau = 0.9$

Figure 21: `GPT-4` generated C-Tests. Colored squares indicate the gap error rates (0.0 1.0)

different strategies, we use the static generation strategy (`STAT`) as a common denominator to compare different C-Tests. Table 17 shows the differences in terms of gap size (i.e., the number of characters) compared to `STAT`. Interestingly, we find that `SIZE` is most similar to `STAT` whereas `MIP` is most dissimilar to `STAT`. Table 18 furthermore shows that `MIP` has the highest overlap (consistently around 50%) to all other strategies in terms of gap placement. Considering that `SIZE` and `MIP` show no significant differences but significantly outperform `SEL` and `GPT-4`, we conclude that we cannot attribute the differences in performance to solely a difference gap size or placement. Moreover, our analysis suggests that C-Tests with different gap sizes and placements can be equally good, highlighting the importance of considering interdependencies between gaps.

	`GPT-4`	`MIP`	`SEL`	`SIZE`
Δ	77	96	0	62
μ	9.6	12.0	0.0	7.8

Table 17: Gap size differences compared to the static generation strategy. We show the total number of differences (Δ) across all eight C-Tests as well as the average difference per C-Test (μ).

	`GPT-4`	`MIP`	`SEL`	`STAT`	`SIZE`
`GPT-4`	100.0	-	-	-	-
`MIP`	48.1	100.0	-	-	-
`SEL`	43.1	51.3	100.0	-	-
`STAT`	40.6	49.4	45.6	100.0	-
`SIZE`	40.6	49.4	45.6	100.0	100.0

Table 18: Overlap of the gap placements when compared to the static (`STAT`) generation strategy (in %). `STAT` and `SIZE` use the same gap placements.

D.5 Importance of Problem Formulation

As discussed in Appendix A.2, the actual problem formulation can make a substantial difference in terms of run time. To investigate if the run time of `MIP` can be further improved, we evaluate two alternative formulations of our optimization objective ($|\tau - \hat{\tau}|$). We especially focus on the absolute value function $|\cdot|$, as this can be done in multiple ways of which some are feasible and others are not.[23] We evaluate three different optimization objectives in this work:

Min/Max Using the minimum and maximum functions: $|\tau - \hat{\tau}| = \max\{\tau, \hat{\tau}\} - \min\{\tau, \hat{\tau}\}$.

Indicator Using indicator constraints (i.e., if-else constructs):

$$|\tau - \hat{\tau}| = \begin{cases} \tau - \hat{\tau}, & \text{if } \tau > \hat{\tau}, \\ \hat{\tau} - \tau, & \text{otherwise.} \end{cases}$$

PWL Using piecewise linear functions:
$|\tau - \hat{\tau}| = a(\cdot) + b(\cdot) + c(\cdot) + d(\cdot)$, with

$$\begin{aligned} a(x) &= \tau \quad , \forall x \leq 0 \\ b(x) &= \tau - \hat{\tau}, \forall 0 < x \leq \tau \\ c(x) &= \hat{\tau} - \tau, \forall \tau < x \leq 1 \\ d(x) &= 1 - \tau, \forall 1 < x. \end{aligned}$$

For evaluation, we repeat the $\texttt{MIP}_{\text{BERT}}$ run time experiments from Appendix D.1.1 using 6 Cores of an *AMD EPYC™ 7742 processor* with 2.25GHz each. To reduce noise from other components in the code (e.g., feature extraction or data loading) we only measure the time required for solving the optimization objective. Table 19 shows the average time (μ) and standard deviation (σ) as well as the minimum (min) and maximum (max) run times for each formulation. Interestingly, we find that the optimization objective has a higher impact on

[23] For instance, $\sqrt{(\tau - \hat{\tau})^2}$ equally describes $|\tau - \hat{\tau}|$ but introduces a quadratic term, resulting in an infeasible model.

Method	μ	σ	max	min
Min/Max	32.02	61.52	656.12	1.33
Indicator	**3.12**	**1.34**	**7.17**	**0.95**
PWL	72.91	370.60	11,481.52	1.15

Table 19: Run time of $\texttt{MIP}_{\text{BERT}}$ using different formulations of our optimization objective (in seconds). We show the average (μ), standard deviation (σ), maximum ($\max$), and minimum ($\min$) run times.

the actual run time than the difference in hardware. For instance, the Min/Max formulation that was used throughout this work has a run time of $\sim$32 seconds; requiring only 10.5 seconds longer on the *AMD* hardware compared to using an *Intel Core™ i5-8400 CPU* with 6 x 2.80GHz (22.5 seconds; cf. Appendix D.1.2). In contrast, using a different optimization objective results in substantially different run times, namely $\sim$3 seconds for the indicator objective and $\sim$73 seconds for PWL. The differences become especially large for the worst-case ($\max$) run times, as the Min/Max formulation requires $\sim$11 minutes and PWL even up to 3 hours and 12 minutes. In contrast, the worst-case run time of the indicator constraint remains low at only $\sim$7 seconds. Overall, the indicator formulation substantially closes the gap of MIP to SIZE and SEL (with a total run time of $\sim$15 seconds) and will be published along with the code to alleviate future research.

D.6 GPT-4 Example

Fig. 22 shows the prompt we use to construct C-Tests using GPT-4. For the few-shot examples, we select the instances from the ACL-2019 dataset with the highest (0.655) and lowest (0.09) difficulty and randomly sample one instance per text (Lee et al. 2019 use four texts in total). Fig. 23 shows the respective response we received after five tries. We regenerated the response if the model produced a C-Test with less than 20 gaps. If the model generated more than 20 gaps, we selected the first 20 gaps. All prompts and responses are provided in the published data.

C-Tests are gap-filling exercises where each only the latter part of a word is made into a gap. A C-Test is generated by placing gaps in an input text. The difficulty of a C-Test is the percentage of errors a student makes across all gaps. Each C-Test consists of 20 gaps. Gaps are indicated by '_' . Here are some examples:
Example 1:
Input Text 1:
The Serge Prokofieff whom we knew in the United States of America was gay, witty, mercurial, full of pranks and bonheur – and very capable as a professional musician. These qualities endeared him to both the musicians and the social-economic haute monde which supported the concert world of the post-World War 1, era.
C-Test 1:
The Se_ Prokofieff wh_ we kn_ in the United States of America was g_ , wi_ , merc_ , full of pra_ a_ bon_ – and very cap_ as a profes_ musi_ . Th_ qual_ ende_ h_ to bo_ the musi_ and the social-economic ha_ mo_ which supported the concert world of the post-World War 1 , era .
Difficulty 1: 0.655
Example 2:
Input Text 2:
It is being fought, moreover, in fairly close correspondence with the predictions of the soothsayers of the think factories. They predicted escalation, and escalation is what we are getting. The biggest nuclear device the United States has exploded measured some 15 megatons, although our B-52s are said to be carrying two 20-megaton bombs apiece.
C-Test 2:
It i_ being fough_ , moreover , i_ fairly clos_ correspondence wit_ the prediction_ of t_ soothsayers o_ the thin_ factories . The_ predicted escalatio_ , and escalatio_ is wha_ we ar_ getting . T_ biggest nuclea_ device t_ United State_ has explode_ measured som_ 15 megatons , although our B-52s are said to be carrying two 20-megaton bombs apiece .
Difficulty 2: 0.09
Example 3:
Input Text 3:
Here was a man with an enormous gift for living as well as thinking. To both persons and ideas he brought the same delighted interest, the same open-minded relish for what was unique in each, the same discriminating sensibility and quicksilver intelligence, the same gallantry of judgment.
C-Test 3:
Here w_ a man w_ an e_ gift f_ living a_ well a_ thinking . T_ both per_ and id_ he bro_ the s_ delighted inte_ , the s_ open-minded relish f_ what w_ unique i_ each , t_ same d_ sensibility a_ quicksilver i_ , the same gallantry of judgment .
Difficulty 3: 0.43
Example 4:
Input Text 4:
The Serge Prokofieff whom we knew in the United States of America was gay, witty, mercurial, full of pranks and bonheur – and very capable as a professional musician. These qualities endeared him to both the musicians and the social-economic haute monde which supported the concert world of the post-World War 1, era.
C-Test 4:
The Se_ Prokofieff wh_ we kn_ in t_ United Sta_ of Ame_ was g_ , witty , merc_ , full o_ pranks a_ bonheur – a_ very cap_ as a profes_ musician . Th_ qualities ende_ him t_ both t_ musicians a_ the social-economic ha_ monde wh_ supported the concert world of the post-World War 1 , era .
Difficulty 4: 0.3425
Example 5:
Input Text 5:
St. Louis sits in the center of a relatively slow-growing and in some places stagnant mid-continent region. Slackened regional demand for St. Louis goods and services reflects the region's relative lack of purchasing power. Not all St. Louis industries, of course, have a market area confined to the immediate neighborhood.
C-Test 5:
St. Louis si_ in t_ center o_ a relatively slow-growing a_ in so_ places stag_ mid-continent region . Slac_ regional dem_ for St. Lo_ goods a_ services refl_ the reg_ 's relative la_ of purch_ power . N_ all St. Lo_ industries , o_ course , ha_ a market ar_ confined t_ the immediate neighborhood .
Difficulty 5: 0.295

Now generate a C-Test with 20 gaps and a difficulty of <τ>
Input Text:

<Text containing all words w>

Figure 22: The prompt used for C-Test generation using GPT-4. We first explain the concept of a C-Test and provide five examples selected from the data collected by Lee et al. (2019). We then add an instruction to generate a C-Test of our target difficulty τ and finally, provide the text passage that contains all words w. All C-Tests used in the study are generated using the same five few-shot examples.

Now generate a C-Test with 20 gaps and a difficulty of 0.1 Input Text:
As little as the act of birth comes into consideration in the whole process and procedure of heredity, just as little is "being-conscious" opposed to the instinctive in any decisive sense; the greater part of the conscious thinking of a philosopher is secretly influenced by his instincts, and forced into definite channels.
C-Test: As litt_ as the a_ of birth com_ into consider_ in the w_ process a_ procedure o_ heredity , jus_ as litt_ is " being - conscious " opp_ to the inst_ in any dec_ sense ; the gre_ part o_ the cons_ thinking of a philo_ is sec_ influenced by h_ instincts , and for_ into def_ channels. Difficulty: 0.1 (Note: This C-Test is generated with a target difficulty of 0.1, but the actual difficulty experienced by a test taker might differ based on their familiarity with the content and context.)

Figure 23: A C-Test generated by GPT-4 for $\mathcal{T}_1$ and $\tau = 0.1$ (last row). Note, that this is the fifth regneration, as previous responses of the model contained less than 20 gaps.

Chapter 8

Efficient Methods for Natural Language Processing: A Survey

Efficient Methods for Natural Language Processing: A Survey

Marcos Treviso[1*], Ji-Ung Lee[2*], Tianchu Ji[3*], Betty van Aken[4], Qingqing Cao[5], Manuel R. Ciosici[6], Michael Hassid[7], Kenneth Heafield[8], Sara Hooker[9], Colin Raffel[10], Pedro H. Martins[1,11], André F. T. Martins[1,11], Jessica Zosa Forde[12], Peter Milder[3], Edwin Simpson[13], Noam Slonim[14], Jesse Dodge[15], Emma Strubell[15,16], Niranjan Balasubramanian[3], Leon Derczynski[5,17], Iryna Gurevych[2], Roy Schwartz[7]

[1]IST/U. of Lisbon and Instituto de Telecomunicações, Portugal,
[2]Technical University of Darmstadt, Germany, [3]Stony Brook University, USA,
[4]Berliner Hochschule für Technik, Germany, [5]University of Washington, USA,
[6]University of Southern California, USA, [7]The Hebrew University of Jerusalem, Israel,
[8]University of Edinburgh, UK, [9]Cohere For AI, USA,
[10]University of North Carolina at Chapel Hill, USA, [11]Unbabel, Portugal, [12]Brown University, USA,
[13]University of Bristol, UK, [14]IBM Research, Israel, [15]Allen Institute for AI, USA,
[16]Carnegie Mellon University, USA, [17]IT University of Copenhagen, Denmark

Abstract

Recent work in natural language processing (NLP) has yielded appealing results from scaling model parameters and training data; however, using only scale to improve performance means that resource consumption also grows. Such resources include data, time, storage, or energy, all of which are naturally limited and unevenly distributed. This motivates research into *efficient* methods that require fewer resources to achieve similar results. This survey synthesizes and relates current methods and findings in efficient NLP. We aim to provide both guidance for conducting NLP under limited resources, and point towards promising research directions for developing more efficient methods.

1 Introduction

Scaling has become a key ingredient in achieving state-of-the-art performance in NLP (Figure 1), as recent research suggests that some capabilities only emerge once models grow beyond a certain size (Wei et al., 2022b). However, despite the merits of scaling, it poses key challenges to making these breakthroughs accessible in resource-constrained environments (Ahmed and Wahed, 2020), in having a non-negligible environmental impact (Strubell et al., 2019; Schwartz et al., 2020a; Derczynski, 2020; Patterson et al., 2021; Wu et al., 2022a), and in complying with hardware constraints (Thompson et al., 2020). To tackle these limitations, there has been renewed focus around research that seeks to improve model *efficiency*.

Definition Efficiency is characterized by the relationship between resources going into a system and its output, with a more efficient system producing the same output with fewer resources. Schwartz et al. (2020a) formalize efficiency as the cost of a model in relation to the results it produces: $\mathrm{Cost}(R) \propto E \cdot D \cdot H$, i.e., the $\mathrm{Cost}(\cdot)$ of producing a certain NLP (R)esult as proportional to three (non-exhaustive) factors: (1) The cost of model execution on a single (E)xample, (2) the size of the (D)ataset, and (3) the number of training runs required for (H)yperparameter tuning. Here we take a different approach, and consider the role that efficiency plays across the different steps in the NLP pipeline, by providing a detailed overview of efficiency methods specific to NLP (Figure 2).

Scope of this Survey We address this work to two groups of readers: (1) Researchers from all fields of NLP working with limited resources; and (2) Researchers interested in improving the state of the art of efficient methods in NLP. Each section concludes with a discussion of limitations, open challenges, and possible future directions of the presented methods. We start by discussing methods to increase *data* efficiency (Section 2), and continue with methods related to *model design* (Section 3). We then consider efficient methods

*Equal contribution. marcos.treviso@tecnico.pt.

Transactions of the Association for Computational Linguistics, vol. 11, pp. 826–860, 2023. https://doi.org/10.1162/tacl_a_00577
Action Editor: Xavier Carreras. Submission batch: 10/2022; Revision batch: 1/2023; Published 7/2023.

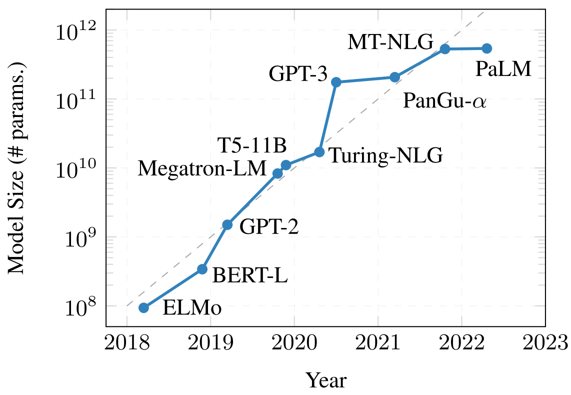


Figure 1: Exponential growth in the number of parameters in pre-trained language models. Adapted from Lakim et al. (2022).

for the two typical training setups in modern NLP: *pre-training* (Section 4) and *fine-tuning* (Section 5). We then discuss methods for making *inference* more efficient (Section 6). While we mainly focus on algorithmic approaches, we provide appropriate pointers regarding *hardware* that are connected to the scale at which we expect to deploy a model (Section 7). We then discuss how to quantify efficiency and what factors to consider during *evaluation* (Section 8), and, finally, how to efficiently decide upon the *best suited model* (Section 9).

To guide the reader, Figure 3 presents a typology of efficient NLP methods considered in this survey.

2 Data

Data efficiency is improved by using fewer training instances, or by making better use of available instances. Fixed compute budgets motivate balancing model size and training data size, especially during pre-training (Hoffmann et al., 2022).

2.1 Filtering

Improving *data quality* can boost performance while reducing training costs during pre-training and fine-tuning. For instance, Lee et al. (2022b) showed that removing duplicates in pre-training increases training efficiency, giving equal or even better model performance compared to using all data. Zhang et al. (2022) used MinhashLSH (Leskovec et al., 2020) to remove duplicates while developing OPT. De-duplication can lead to substantially reduced computation cost, especially in cases with abundant pre-training data but limited compute budget (Hoffmann et al., 2022).

Similar observations have been made for fine-tuning. For instance, Mishra and Sachdeva (2020) found—via adversarial filtering (Zellers et al., 2018)—a subset of only ~2% of the SNLI data (Bowman et al., 2015) that leads to performance comparable to using the full corpus. While such filtering approaches are useful for mitigating biases (Le Bras et al., 2020), they may not always serve as tools to filter existing datasets, as these often suffer from insufficient training data.

2.2 Active Learning

Active learning aims to reduce the number of training instances. In contrast to filtering, it is applied during data collection (instead of after) to only annotate the most helpful or useful instances for training (Settles, 2012; Ren et al., 2021b). To assess usefulness of an instance without knowing its actual label, one can use the model *uncertainty*—assuming that labeling instances with the highest uncertainty is most helpful (Lewis and Gale, 1994; Tang et al., 2002; Gal et al., 2017; Yuan et al., 2020); instance *representativeness*—to maximize diversity of sampled instances while avoiding outliers (Bodó et al., 2011; Sener and Savarese, 2018; Gissin and Shalev-Shwartz, 2019); or a combination of both criteria (Kirsch et al., 2019; Ash et al., 2020; Margatina et al., 2021; Siddiqui et al., 2021; Agarwal et al., 2022). Active learning has been successfully applied in machine translation (MT; Liu et al. 2018), language learning (Lee et al., 2020), entity linking (Klie et al., 2020), and coreference resolution (Li et al., 2020a; Yuan et al., 2022). Despite its advantages, some open questions make active learning difficult to apply in practice. It remains unclear how model-based sampling impacts the performance of models using architectures different from that in sampling (Lowell et al., 2019; Ein-Dor et al., 2020). Also, selecting ‘‘difficult’’ instances may increase annotation cost and difficulty (Settles et al., 2008; Lee et al., 2022a). Finally, it is prone to selection biases and can favor outliers (Cortes et al., 2008; Karamcheti et al., 2021).

2.3 Curriculum Learning

Curriculum learning aims to find a data ordering that reduces the number of training steps required to achieve a target performance (Elman,

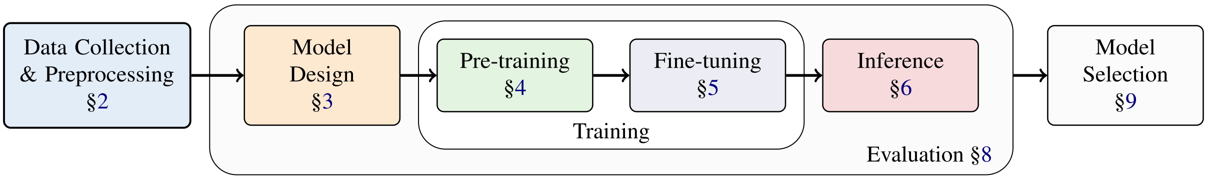


Figure 2: Schematic overview of the efficient NLP stages covered in this paper, starting with data collection and model design, followed by training and inference, and ending with evaluation and model selection. Notably, the training stage is divided into two parts: pre-training, which aims to learn generalizable parameters, and fine-tuning, which optimizes these parameters for specific downstream tasks.

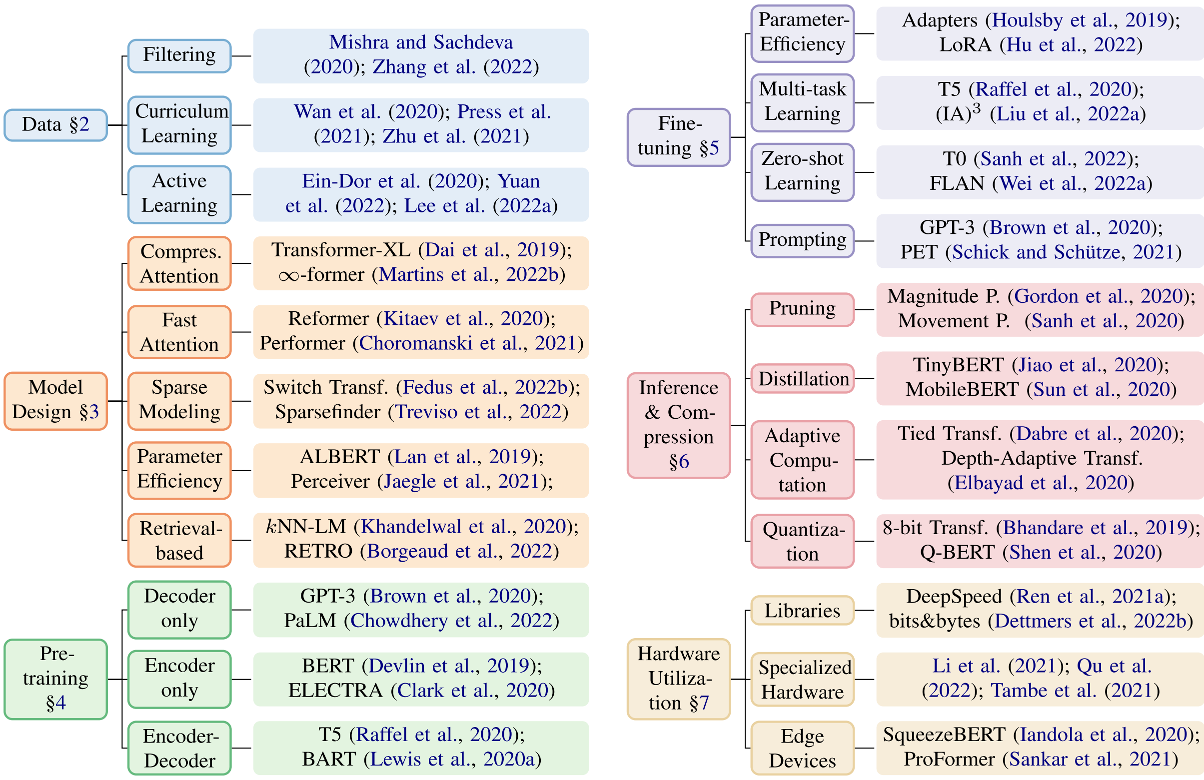


Figure 3: Typology of efficient NLP methods.

1993; Bengio et al., 2009). This method does not reduce dataset size, but does improve its utilization. Hence, it is a common approach for improving training efficiency in both pre-training and fine-tuning. Many curriculum learning methods order instances by difficulty, using heuristics such as sentence length. This has yielded improvements for transformer pre-training (Press et al., 2021; Agrawal et al., 2021) as well as fine-tuning on tasks such as question answering (Tay et al., 2019), MT (Zhang et al., 2019), and others (Xu et al., 2020).

A major challenge in curriculum learning is determining *pace*, i.e., when to progress to more difficult instances. If not chosen carefully, curriculum learning can waste compute on ''easy'' instances. To tackle this, work has investigated adaptive ordering strategies based on current model state, called *self-paced learning* (Kumar et al., 2010). This has been successfully applied to improve performance in MT using model and data uncertainty (Wan et al., 2020; Zhou et al., 2020; Zhao et al., 2020), and in dialog generation with knowledge distillation (Zhu et al., 2021). However, self-paced learning involves large training costs, and disentangling instance ordering from factors such as optimizer choice and batch size is non-trivial (Dodge et al., 2020).

2.4 Estimating Data Quality

In an era of ever larger datasets, auditing and estimating the quality of data is increasingly challenging. Datasets frequently present high levels of noise and misaligned instances (Kreutzer et al., 2022). Estimating data quality encompasses research efforts which propose better uncertainty estimates (Baldock et al., 2021; D'souza et al., 2021; Ethayarajh et al., 2022) as well as analytical tools such as dataset cartography (Swayamdipta et al., 2020). Qualitative tools include documentation for datasets and model attributes (Gebru et al., 2021).

3 Model Design

Efficient model design covers architectural changes and adding new modules to accelerate training.

3.1 Improving Attention in Transformers

The transformer's self-attention mechanism has a quadratic dependency on sequence length which is not fully utilized by existing models (Hassid et al., 2022). To reduce computational costs, efficient attention mechanisms for long sequences have been proposed (Tay et al., 2022). Existing strategies include better using already-processed segments via recurrence to connect multiple segments (Dai et al., 2019), learning a network to compress a longer-term memory (Rae et al., 2020), separately modeling global and local attention (Ainslie et al., 2020), and modeling long inputs as a continuous-time signal (Martins et al., 2022b). Another line of research uses fixed attention patterns, where tokens attend to their immediate context (local attention) and possibly to a few global positions (global attention; Beltagy et al., 2020; Zaheer et al., 2020; Child et al., 2019). Compared to using the full self-attention matrix, such approaches can scale linearly with the input length.

Some methods learn attention sparsity patterns directly from data, e.g., by grouping tokens into buckets, leading to a more accurate yet more expensive approximation of the full attention matrix (Kitaev et al., 2020; Daras et al., 2020; Roy et al., 2021). Instead of seeking better attention patterns, some strategies modify the attention *mechanism* and derive low-rank approximations to the query-key matrices via reverse application of the kernel trick, resulting in linear time attention (Katharopoulos et al., 2020; Choromanski et al., 2021; Peng et al., 2020; Zhai et al., 2021). Recently, IO-aware attention mechanisms have been proposed, decreasing reads and writes to the attention matrix to GPU high-bandwidth memory (Dao et al., 2022b).

Despite various improvements in attention mechanisms, most of them struggle with very long sequences (Tay et al., 2021). S4 (Gu et al., 2022b), and its successors (Gupta et al., 2022; Mehta et al., 2023; Gu et al., 2022a), suggest an alternative to transformers that alleviates the short memory problem and the quadratic bottleneck cost of self-attention by discretizing state space representations through parameterization of the state matrix. More recently, Mega (Ma et al., 2023) replaced the multi-headed transformer attention mechanism with a single-headed mechanism that receives contextualized vectors from a multidimensional exponential moving average module, and then splits the input into multiple fixed-length chunks to reduce the computation cost. Both S4 and Mega strongly outperform attention-based methods on all tasks of the Long Range Arena benchmark (Tay et al., 2021), while increasing training speed by approximately 5x and reducing memory cost by about 15% when compared to a standard transformer. This success is attributed to their convolutional structure, which emphasizes nearby tokens and has a parameter count that grows sub-linearly with sequence length (Li et al., 2022b).

3.2 Sparse Modeling

To leverage sparsity for efficiency, many models follow the mixture-of-experts (MoE) concept (Jacobs et al., 1991; Shazeer et al., 2017; Fedus et al., 2022a), which routes computation through small subnetworks instead of passing the input through the entire model. Relevant works on this line include GShard (Lepikhin et al., 2021), Switch Transformer (Fedus et al., 2022b), and ST-MoE (Zoph et al., 2022), which replace the feed-forward layers in transformers with MoE layers. More recently, Rajbhandari et al. (2022) scaled transformers up by compressing and optimizing the usage of MoE. Overall, MoE models have been shown to achieve strong performance across several NLP tasks while reducing the overall resource consumption (Section 8). For instance, GLaM (Du et al., 2022) used only $\sim\frac{1}{3}$

of GPT-3's energy consumption (with additional hardware-based optimization), while Rajbhandari et al. (2022) reached a 5x reduction in terms of training cost. However, MoE models have also exhibited training instabilities in practice, and may require architecture-specific implementation (Zoph et al., 2022; Mustafa et al., 2022).

Another promising direction for exploiting sparse modeling is Sparsefinder (Treviso et al., 2022), which extends the Adaptively Sparse Transformer (Correia et al., 2019) to allow a more efficient attention mechanism by identifying beforehand the sparsity pattern returned by entmax attention—a sparse alternative to (dense) softmax attention (Peters et al., 2019). Finally, sparsity can also be induced via modularity, e.g., by encapsulating task-specific parameters (Ponti et al., 2022).

3.3 Parameter Efficiency

Methods that reduce parameter count can reduce computational costs and memory usage. One such approach is to share weights across layers of a model while maintaining the downstream task performance (Dehghani et al., 2019; Lan et al., 2019). Besides sharing weights, Perceiver (Jaegle et al., 2021) also minimizes the computational cost of self-attention on long sequences by mapping the input to a small latent vector. ALBERT (Lan et al., 2019) further uses matrix decomposition to reduce the size of the embedding layer, which is one of the largest consumers of model parameters. Finally, Reid et al. (2021) studied ways to share weights in transformers, showing that sharing only the middle layers of the model outperforms the alternatives.

3.4 Retrieval-Augmented Models

Parametric models can be combined with retrieval mechanisms for text generation, leading to semi-parametric models (Gu et al., 2018; Lewis et al., 2020b; Li et al., 2022a). This typically amounts to trading model size with the number of database entries. For instance, RETRO (Borgeaud et al., 2022) matched the performance of models 25 times larger by retrieving chunks of tokens from a 2 trillion token database. At inference time, the model retrieves tokens / phrases / sentences from a database, which are used by the model through a combination of probability distributions (Khandelwal et al., 2020), gating mechanisms (Yogatama et al., 2021), or attention (Borgeaud Borgeaud et al., 2022).

These models also have good generalization properties: By retrieving from domain-specific databases, they can be applied to new domains, reducing the need for domain-specific fine-tuning (Khandelwal et al., 2020, 2021). That is, having an explicit ''memory'' also allows retrieval-augmented models to be adapted post-training. Although they may yield slow running speeds since the retrieval time grows as the datastore scales, recent work proposed strategies to alleviate this, such as pruning the database (He et al., 2021), having smaller input-dependent databases (Meng et al., 2022), reducing the representation dimension (Martins et al., 2022a), and clustering data points (Wang et al., 2021b; Alon et al., 2022). In particular, Martins et al. (2022c) have shown that carefully constructing a database not only leads to better translations than fine-tuning, but can also reduce the total translation time (inference + online adaptation).

3.5 Model Design Considerations

Despite considerable advances, one major challenge is modeling long sequences in many real-world documents. For instance, sustainability reports have on average 243.5 pages (Manes-Rossi et al., 2018), which substantially exceeds the maximum length (16k tokens) found in Path-X from Long Range Arena (Tay et al., 2021). In fact, the ability of a model to handle longer sequences than those seen during training may depend on design choices, such as the attention mechanism (Dubois et al., 2020) and the positional encoding (Shaw et al., 2018; Press et al., 2022). The effect of this behavior when using transformers with sub-quadratic attention, sparse modeling approaches, or parameter efficient models is not yet well understood.

While sparse modeling approaches like MoE can substantially reduce inference and training costs, they require additional model parameters for retraining specialized modules and have instability issues during training (Zoph et al., 2022). Models that rely on built-in sparse transformations, such as entmax (Peters et al., 2019), have achieved strong results without stability issues, but have not yet fully realized competitive efficiency gains. Combining MoE with built-in sparse functions may be a promising research direction, e.g., by using entmax in the routing layer.

In retrieval-augmented models, the quality of the retrieval component is critical to performance, and the tradeoff between storing information in model parameters vs. external resources needs to be better understood, especially when deploying models in low-resource settings like edge devices. Finally, while new model designs improve efficiency through different means, further improvements can emerge from combining approaches, such as making MoE more efficient using quantization (Section 6.3) and using parameter-efficient models for distillation (Section 6.2).

4 Pre-training

Modern transfer learning approaches in NLP typically involve *pre-training* a model in a self-supervised fashion on large amounts of text before fine-tuning it on specific tasks (Section 5). Improving the pre-training procedure of a model can significantly reduce the cost of hyperparameter tuning and increase data efficiency for fine-tuning (Peters et al., 2018; He et al., 2019; Neyshabur et al., 2020).

4.1 Optimization Objective

The choice of the task can determine the success of the pre-trained model on downstream tasks. Left-to-right language models, such as GPT (Radford et al., 2019; Brown et al., 2020) and PaLM (Chowdhery et al., 2022), are trained with the *causal language modeling* (CLM) objective, which involves predicting the next token given a context. BERT (Devlin et al., 2019) uses a *masked language model* (MLM) task, which involves filling randomly masked tokens.

To make better use of available data, various masking strategies have been investigated. Masking objects and content words only rather than random tokens (Bitton et al., 2021), or masking more tokens (Wettig et al., 2022), has led to higher task performance and more efficient use of the available data. ELECTRA (Clark et al., 2020) and DeBERTa (He et al., 2023) tried *replaced token detection* (RTD), an objective that uses a small generator model to replace input tokens, and converges more quickly to better performance. A limitation of the MLM and RTD objectives is that they work with single token replacements. T5 (Raffel et al., 2020) and BART (Lewis et al., 2020a) overcome this by adopting a denoising *sequence-to-sequence* objective to pre-train an encoder-decoder model, allowing the decoder to predict a span of tokens for masked positions. In practice, this allows training on shorter sequences without losing task performance, which helps to reduce training costs.

4.2 Pre-training Considerations

Despite increases in the size of pre-trained models (cf. Figure 1), many pre-training efficiency gains come from improving model design (Section 3) and selection (Section 9) as well as making more efficient use of the available data (Section 2). These factors have had a greater impact on model performance than the pre-training objective itself (Alajrami and Aletras, 2022). However, pre-training is usually computationally expensive, requiring significant amounts of GPU memory and computational power (Rae et al., 2021), and may require large amounts of quality data, which can be difficult to acquire and curate (Kaplan et al., 2020). Surprisingly, as demonstrated by Chinchilla (Hoffmann et al., 2022), decreasing model size to account for the amount of available data not only leads to better performance, but also reduces computational cost and improves model applicability to downstream tasks. Continued focus on the role of data in efficient pre-training is a promising direction, such as recent work studying the role of (de-)duplication of examples in large-scale pre-training corpora (Lee et al., 2022b). While transformers have been the dominant architecture in pre-trained models, more efficient modeling methods such as state space representations and MoEs (Section 3.1) have the potential to overcome some challenges of pre-training transformers.

5 Fine-tuning

Fine-tuning refers to adapting a pre-trained model to a new downstream task. While some approaches explicitly aim to make the fine-tuning process more efficient, in this survey, we use a broader definition of fine-tuning that includes any method used to apply a pre-trained model to a downstream task.

5.1 Parameter-Efficient Fine-Tuning

Gradient-based fine-tuning typically involves training all model parameters on a downstream task. Hence, fine-tuning a pre-trained model on a new task creates an entirely new set of model parameters. If a model is fine-tuned on many tasks, the storage requirements can become onerous. Adapting a pre-trained model to downstream tasks by training a new classification layer and leaving the rest of the parameters fixed (a.k.a. feature extraction; Peters et al., 2018) updates dramatically fewer parameters than training the full model but has been shown to produce worse performance and has become less common (Devlin et al., 2019).

Several approaches have been proposed to adapt a model to a new task while only updating or adding a relatively small number of parameters—up to four orders of magnitude fewer parameters than full-model fine-tuning—without sacrificing (and in some cases improving) performance. Adapters (Houlsby et al., 2019; Bapna and Firat, 2019; Rebuffi et al., 2017; Pfeiffer et al., 2020) inject new trainable dense layers into a pre-trained model, while leaving the original model parameters fixed. They have recently been improved by the Compacter method (Karimi Mahabadi et al., 2021), which constructs the adapter parameter matrices through Kronecker products of low-rank matrices. While adapters can reduce training time due to a reduced number of trained parameters, and mitigate some deployment costs due to reduced storage requirements, one shortcoming is increased inference time due to more parameters (Rücklé et al., 2021). To mitigate this, Moosavi et al. (2022) proposed training an additional layer selector to only use adapter layers necessary for a given task.

As an alternative to adding new layers, parameter-efficiency can be achieved by directly modifying activations with learned vectors, either by concatenation (Liu et al., 2021a; Li and Liang, 2021; Lester et al., 2021), multiplication (Liu et al., 2022a), or addition (Ben Zaken et al., 2022). Two notable approaches are prefix-tuning (Li and Liang, 2021) and prompt-tuning (Lester et al., 2021), which fine-tune continuous prompts as an alternative to engineering discrete prompts (cf. Section 5.3). Although they are conceptually similar to adapters, He et al. (2022b) show that they are equivalent to a parallel insertion, whereas adapters are inserted sequentially. Alternatively, rather than adding new parameters or changing the computational graph, it is possible to make sparse (Sung et al., 2021; Guo et al., 2021) or low-rank (LoRA, Hu et al., 2022) updates. Finally, optimization can be performed in a low-dimensional subspace (Li et al., 2018), which leads to parameter-efficient updates (Aghajanyan et al., 2021b). Although low-rank approaches mitigate the issue of increased inference time, they require an additional optimization step to identify the best rank. To mitigate this, Valipour et al. (2022) proposed a dynamic solution that substantially reduces training time compared to LoRA. Lastly, Wang et al. (2022b) devised AdaMix to combine different parameter efficient fine-tuning techniques together via routing and showed that their approach can even outperform full fine-tuning.

5.2 Multi-Task and Zero-Shot Learning

While traditional transfer learning includes fine-tuning, there are other paradigms that allow for immediate application of a pre-trained model to a downstream task of interest. *Multi-task learning* (Caruana, 1997; Ruder, 2017) aims to train a single model that can perform a wide variety of tasks out of the box. Typically, this is done by fine-tuning on data from all downstream tasks of interest. Multi-task models can improve fine-tuning performance (Raffel et al., 2020; Aghajanyan et al., 2021a; Aribandi et al., 2022; Liu et al., 2022a). In certain cases, a multi-task model works on new tasks without any fine-tuning, also referred to as *zero-shot generalization* (Sanh et al., 2022; Wei et al., 2022a). Radford et al. (2017, 2019) and Brown et al. (2020) demonstrated that language models trained with an unsupervised objective can perform a variety of tasks out-of-the-box. While it can circumvent the need for fine-tuning, zero-shot ability depends on model size and only becomes competitive at a certain scale (Wei et al., 2022b).

5.3 Prompting

Inspired by models like GPT-3 (Brown et al., 2020), prompting refers to casting a task as a textual instruction to a language model (Liu et al., 2023). In general, prompts can be either crafted manually or automatically using fill-in templates for token, span, and sentence-level completion (Petroni et al., 2019; Brown et al., 2020; Shin et al., 2020). This makes prompting applicable to more challenging NLP tasks, such as QA, MT, and summarization (Schick and Schütze, 2021). Although prompting eliminates the need for any fine-tuning, identifying good prompts can be difficult (Liu et al., 2021a). Hence, recent work investigates the automated creation of suitable prompts, albeit with additional training cost (Bach et al., 2022).

5.4 Fine-Tuning Considerations

An emerging problem with large language models is the universally high cost of fully fine-tuning them (Chen et al., 2021). Although prompting (without fine-tuning) can alleviate this issue, designing prompts can be tedious—even with automated help. One promising direction for efficiently introducing new knowledge into models is to combine existing methods for efficient fine-tuning. This could involve methods such as that used by Karimi Mahabadi et al. (2022), who proposed task-specific adapters to avoid generating prompts, and achieved considerable speed ups while tuning under 1% of parameters. Another challenge in adopting large pre-trained models for fine-tuning is the complexity in interpreting the final model, due in part to the use transformers. To gain a better understanding of these models while still leveraging efficiency, a promising direction is to combine techniques such as sparse modeling and parameter-efficient methods (Correia et al., 2019; Treviso et al., 2022).

6 Inference and Compression

Inference involves computing a trained model's prediction for a given input. Inference can be made more efficient by accelerating the process for time efficiency (latency), or by compressing the model to reduce memory requirements.

6.1 Pruning

Proposed by LeCun et al. (1989), pruning removes irrelevant weights from a neural network to reduce computation, and furthermore, decreases memory capacity and bandwidth requirements. Pruning can be applied at different stages of the NLP pipeline (Figure 2). For instance, Gordon et al. (2020) found that up to $\sim$40% of BERT can be pruned at pre-training without affecting its performance. Others proposed pruning methods that work as regularizers and can be applied to pre-training and fine-tuning (Louizos et al., 2018; Wang et al., 2020b). Finally, work has investigated pruning during fine-tuning (Han et al., 2015; Sanh et al., 2020) or dynamically during inference (Fan et al., 2020).

Pruning was initially introduced at the individual weight level (unstructured pruning), but more recent approaches prune larger components of the network (structured pruning). Examples of the latter include removing attention heads (Voita et al., 2019; Michel et al., 2019), weak attention values (Ji et al., 2021; Qu et al., 2022), and even entire hidden layers (Dong et al., 2017; Sajjad et al., 2023). In particular, Xia et al. (2022) found that pruning all these components yields more accurate and efficient models. When comparing the two pruning approaches, unstructured pruning is often found to better preserve a model's performance (Gale et al., 2019; Ahia et al., 2021), but existing hardware often cannot exploit the resulting sparsity. In contrast, structured pruning methods often lead to a higher improvement in terms of inference speed (Hoefler et al., 2021). The increasing popularity of pruning methods has further raised the question of how to quantify and compare them (Gale et al., 2019; Blalock et al., 2020; Tessera et al., 2021; Hoefler et al., 2021) and motivated work that combines pruning with other efficiency methods such as adapters (Rücklé et al., 2021) and distillation (Zafrir et al., 2021).

While early pruning (e.g., during pre-training) can further reduce training costs, it increases the risk of over-pruning: removing nodes essential for downstream task performance (Gordon et al., 2020). Although this can be mitigated by ''regrowing'' pruned weights (Mostafa and Wang, 2019), this increases training costs. Other pruning downsides include additional costs for hyperparameter tuning such as the number of preserved weights.

6.2 Knowledge Distillation

The process of knowledge distillation uses supervision signals from a large (teacher) model to train a smaller (student) model (Hinton et al., 2015), and often leads to the student outperforming a similarly sized model trained without this supervision. While early work focused on distilling task-specific models (Kim and Rush, 2016), recent work focuses on distilling pre-trained models that can then be fine-tuned on specific downstream tasks (Sanh et al., 2019; Liu et al., 2020; Jiao et al., 2020; Sun et al., 2020; Gou et al., 2021). The downsides of distillation include the added cost of tuning student hyperparameters and the potential for reduced performance and generalization capability (Stanton et al., 2021). Recently, Zhu et al. (2022) discovered that some performance loss is due to undistillable classes and suggested ways to address this.

6.3 Quantization

Mapping high-precision data types to low-precision ones is referred to as *quantization*. Quantization can be applied at different stages in the NLP model-building pipeline to reduce training and inference costs. Various research has shown that low-precision data format can reduce memory consumption by 4x–24x and improve the throughput by 4.5x compared to 32-bit floating point format. Various studies targeted specific precision-levels such as integers (Kim et al., 2021), 8-bit (Quinn and Ballesteros, 2018; Zafrir et al., 2019; Bhandare et al., 2019; Prato et al., 2020; Dettmers et al., 2022a), ternary (Zhang et al., 2020; Ji et al., 2021; Zadeh et al., 2022), and even binary representations (Bai et al., 2021).

Different components may have different sensitivities regarding their underlying precision, so there is a body of work on mixed-precision quantization. Shen et al. (2020) showed that embedding layers require more precise parameter representations than the attention layer, while Kim et al. (2021) showed that nonlinear functions require more bits than the general matrix multiplication. Others defined quantization as a constrained optimization problem to automatically identify layers where lower precision is sufficient (Hubara et al., 2021). Finally, several studies proposed quantization during training to make them robust against performance loss after quantization (Zafrir et al., 2019; Kim et al., 2021; Stock et al., 2021). For instance, Bai et al. (2021) and Zhang et al. (2020) proposed using knowledge distillation to maintain the accuracy of binarized and ternarized models. These show that component-customized quantization can preserve accuracy while improving efficiency. To maximize the benefit from quantization, one should also consider the available underlying hardware and associated specialized kernels compatible with different bit representations (Noune et al., 2022; Kuzmin et al., 2022).

6.4 Inference Considerations

While efficiency during pre-training and fine-tuning focuses on the computational resources and time required to train and optimize a model, inference efficiency is focused on how well a learned model can perform on new input data in real-world scenarios. Moreover, inference optimization is ultimately context-specific and the requirements vary according to the use-case. Therefore, there is no one-size-fits-all solution to optimizing inference, but instead a plethora of techniques. For instance, while Wu et al. (2022b) combine several methods to achieve utmost model compression, other works improve task-specific mechanisms such as beam-search in MT (Peters and Martins, 2021). Parallelism can also be leveraged to increase inference efficiency, but its effectiveness may depend on the hardware available (Rajbhandari et al., 2022). Dynamic computation techniques, such as early-exit (Schwartz et al., 2020b; Xin et al., 2020) and MoE (Section 3.1), can improve inference efficiency by selectively performing computation only on the parts of the model that are needed for a given input. However, current dynamic computation methods often use eager execution mode, which can prevent them from low-level optimization, as noted by Xu and McAuley (2023). Work focusing on inference efficiency should carefully report the exact target setting (hardware, eager vs. static execution framework). Accordingly, promising directions for optimizing inference efficiency might consider tighter integration across or more general purpose approaches with respect to algorithm, software, and hardware. One recent such example is neural architecture search for hardware-specific efficient transformers (Wang et al., 2020a).

7 Hardware Utilization

Many hardware-specific methods focus on reducing GPU memory consumption, a major bottleneck in transformer models. Others leverage specialized hardware, co-design of hardware, and adaptations targeted to edge devices. Many techniques can be combined and applied across different stages of training and inference (Figure 2) for further efficiency.

7.1 Reducing Optimizer Memory

Optimizers that track gradient history incur a memory cost. Libraries like DeepSpeed (Ren et al., 2021a) allow gradient history to be offloaded from GPU to CPU RAM where computation is performed via efficient AVX instructions. `bitsandbytes` (Dettmers et al., 2022b) uses dynamic block-wise quantization to reduce memory pressure. It splits tensors into blocks and quantizes each block individually. This reduces memory consumption by 75% and improves training times due to reduced inter-GPU communication.

7.2 Specialized Hardware

Specialized NLP hardware has been built using Application Specific Integrated Circuits or Field Programmable Gate Arrays, though it is not yet broadly available. These designs use dedicated units for efficient operations like quantization and pruning (Section 6). For example, Zadeh et al. (2020, 2022), Li et al. (2021), and Qu et al. (2022) support ultra-low-bit and mixed precision computation that cannot be done on CPUs/GPUs; Ham et al. (2020, 2021) and Wang et al. (2021a) design hardware that predicts and prunes redundant heads/tokens and weak attention values in transformers. Qu et al. (2022) present a design that balances the workload to alleviate the irregularity in the pruned attention. Others develop new types of processors and memories optimized for transformer components: Lu et al. (2020) and Liu et al. (2021b) implemented dedicated hardware for softmax and layer normalization respectively, and Tambe et al. (2021) used embedded Resistive RAM—a nonvolatile memory with low latency and energy consumption—to store word embeddings.

7.3 Co-design

Some work optimizes hardware, software, and algorithms jointly, which historically has been a common way to realize efficiency gains (Hooker, 2021). For instance, Lepikhin et al. (2021) demonstrated that improving the underlying compiler can substantially improve parallelization and enable scaling. Other examples for co-design focus on hardware-aware mixture of experts models and attention mechanisms to produce substantial speedups (He et al., 2022a; Rajbhandari et al., 2022; Dao et al., 2022b). Barham et al. (2022) proposed a gang-scheduling approach with parallel asynchronous dispatch that leads to substantial efficiency gains. Finally, Hinton (2022) suggested ‘‘mortal computation’’, an extreme form of co-design, where by training a model that is tailored to a specific hardware, the need to guarantee consistent software behavior across different hardware is reduced, potentially saving computation.

7.4 Edge Devices

Tight compute and memory constraints on edge devices motivate a separate set of efficiency solutions. SqueezeBERT (Iandola et al., 2020) incorporates group convolutions into self-attention to improve efficiency on mobile devices. EdgeFormer (Ge et al., 2022) interleaves self-attention layers with lightweight feed-forward layers and an encoder-heavy parameterization to meet edge memory budgets. GhostBERT (Huang et al., 2021) uses *ghost* modules built on depth-wise separable convolutions used in MobileNets (Howard et al., 2017). LiteTransformer (Wu et al., 2020) uses long-short range attention to encode local context by convolutions for MT in resource-constrained settings. Through quantization `llama.cpp`[1] runs a 7B-parameter LLM on recent mobile phone hardware. Finally, ProFormer (Sankar et al., 2021) reduces runtime and memory via locality sensitive hashing and local projection attention layers.

7.5 Hardware Considerations

To deliver more computational power, vendors pack denser computational units into domain-specific hardware, such as tensor cores in Intel FPGAs, Xilinx AI Engines, and matrix processors in the Google TPU. However, irregularities in the transformer, like sparsity and mixed data types, restrict the use of these resources. We suggest focusing on adapting efficient transformers to existing specialized hardware platforms, including using hardware-optimized data formats like block floating point, and exploring sparsity on dense tensor units.

8 Evaluating Efficiency

Evaluating efficiency requires establishing which computational aspect one aims to minimize. We discuss the two most prominent aspects (FLOP/s and power consumption), and list open challenges.

8.1 Evaluation Measures

Pareto Optimality When improving efficiency, multiple factors often need to be traded off. For instance, longer training time can increase task performance, but simultaneously increase

[1] `https://github.com/ggerganov/llama.cpp`, 20 March 2023.

resource consumption. A principled way to characterize trade-offs is to identify Pareto-optimal solutions (Pareto, 1896), those for which no other system reaches a better or equal task performance with lower resource consumption. As there may be several Pareto-optimal solutions, final choice depends on the application context; a small, average-quality model and a large, higher-quality model can both be optimal. Thus, as long as a model contributes to or extends the Pareto-optimal curve for a given problem and measurement space, it it worthwhile—even if other solutions may use less resources or produce higher quality scores.

Advancing NLP by pushing Pareto barriers is an established practice (Kim et al., 2019; Bogoychev et al., 2020; Behnke and Heafield, 2021). For instance, the WNGT 2020 MT shared task (Birch et al., 2020) considers the Pareto frontier between real time taken, system or GPU memory usage, and model size, as well as BLEU score. Puvis de Chavannes et al. (2021) included power consumption as a trade-off against perplexity to explore Pareto-efficient hyperparameter combinations for transformer models. Finally, Liu et al. (2022b) examined Pareto efficiency for a number of tasks in an attempt to narrow model selection search space.

FLOP/s A frequently reported efficiency measure is the number of floating point operations (FLOPs) and floating points per second (FLOP/s). While these discrete metrics seem well defined in terms of what the hardware does, there is some variation at multiple stages of the stack, adding uncertainty. For example, different operations may count as a FLOP on different hardware; non-floating-point operations are not considered; and hardware is rarely 100% utilized and achieving this productively is a challenge, so theoretical FLOP/s performance cannot be multiplied with time elapsed to yield the amount of computing performed. Still, FLOP/s per unit power can indicate which hardware choices have the potential to offer Pareto-efficient trade-offs (Hsu et al., 2005).

Power Consumption There exist various ways to measure power consumption, for instance, by using specific hardware such as an electricity meter. While this can provide precise figures with a high temporal accuracy, it cannot provide a fine-grained estimate for individual computers in a network. Moreover, it does not cover external energy costs such as cooling or networking. Another way is to use software tools such as MLCO2 (Luccioni et al., 2019). Some tools even provide a real-time breakdown of the power consumption of different components within a machine (Henderson et al., 2020) or local machine API-reported figures to stop training early if prudent (Anthony et al., 2020). Finally, Hershcovich et al. (2022) introduced a model card for NLP systems that encourages researchers to document efficiency in a consistent manner.

Measuring power consumption programmatically comes with a number of caveats. First, sampling frequency is often restricted at various levels of the stack and may result in a lag in measurement start. Consequently, shorter experiments may log an energy use of zero, and there will almost always be energy demand that is missed. Second, inefficiencies such as heat loss are not reported by current APIs and hence do not cover cooling and other system management activities. Third, not all architectures and operating systems are supported. For instance, power consumption under macOS is difficult to manage, and direct figures for TPU power consumption are not available.

Carbon Emissions Carbon emissions are usually computed using the power consumption and the carbon intensity of the marginal energy generation used to run the program. Thus, low-energy does not mean low-carbon, and high-energy models can—in the right region and with some care—be zero-carbon in terms of point energy consumption impact, if executed at the right time (i.e., when the energy mix is low-carbon intensity; Dodge et al., 2022). For estimating the CO_2 emissions from a specific program execution, APIs such as ElectricityMap[2] provide real-time access to carbon intensity for many regions. However, as carbon intensity varies and is affected by other factors like the power usage efficiency in a data center, it is often a poor basis for comparison; in fact, Henderson et al. (2020) recommended using multiple runs for a stable estimate. Furthermore, one needs to consider that zero-carbon program executions still consume energy, and that efficiency does not intrinsically guarantee a reduction in overall resource consumption, as the resulting cost reduction may lead to an increase

[2] `https://electricitymap.org`.

in demand counteracting any gains, an effect known as Jevons' paradox (Jevons, 1866).

8.2 Open Challenges in Measuring Efficiency

Hardware choice can lead to pronounced differences in certain efficiency measurements such as latency and thoroughput (Lee-Thorp et al., 2022). Properly measuring efficiency remains a major challenge (Cao et al., 2020).

Separating Different Stages It is important to characterize efficiency of pre-training and fine-tuning stages separately (Sections 4 and 5). Models may present different memory requirements during training yet result in trained models with comparable inference memory consumption. This is because training often involves design choices that increase the memory overhead of backward propagation. Further, some optimizers may require substantially more memory than others. Similarly, parameter sharing techniques may show few benefits during training but show memory improvements at inference (Dehghani et al., 2022). Finally, while larger models run more slowly than smaller ones, they converge faster and better compress using methods like pruning and quantization (Li et al., 2020c).

Disagreement Between Cost Factors As partially discussed in Section 7.2, cost indicators may disagree with each other. For instance, MoEs increase the overall parameter count, but improve the trade-off between quality and FLOPs, as they minimize the per-data cost by routing to subsections of the model (Rajbhandari et al., 2022). Conversely, unstructured sparsity techniques can significantly minimize the overall number of FLOPs, yet in practice, they introduce low-level operations that can lead to far higher memory requirements to store the indices that indicate what part of the matrix is sparse (Qu et al., 2022). Finally, Chen et al. (2022) and Dao et al. (2022a) found specific sparsity patterns that achieve more predictable speedups with current hardware.

Trade-offs with Other Desiderata One major, but seldom studied, concern when improving efficiency are trade-offs with other desiderata such as fairness and robustness. For instance, Hooker et al. (2020), Renduchintala et al. (2021), and Silva et al. (2021) found that compression techniques such as pruning can amplify existing biases; Mohammadshahi et al. (2022) and Ogueji et al. (2022) further explored these trade-offs in a multilingual setting. So far, only a few studies investigated preserving a model's fairness when increasing its efficiency. To quantify such effects, Xu et al. (2021) proposed a novel metric called loyalty, which measures the resemblance of predicted distributions made by teacher and student models. Hessenthaler et al. (2022) established that many approaches for increasing fairness in NLP models also increase computation, and jointly with work like Wang et al. (2022a) showed that distillation can decrease model fairness. Xu and Hu (2022) studied these effects more systematically, with mixed conclusions. While more positive insights have been found with respect to other desiderata such as out-of-distribution (OOD) generalization (Ahia et al., 2021; Iofinova et al., 2022; Ogueji et al., 2022) and model transfer (Gordon et al., 2020), more work is needed to better understand and benchmark the impact of efficiency beyond accuracy.

9 Model Selection

Finally, we discuss lines of research that opt to efficiently select a well-performing model variant.

9.1 Hyperparameter Search

The performance of machine learning methods can be improved by choosing hyperparameters carefully. Model-based techniques such as Bayesian optimization (BO; Snoek et al., 2012; Feurer et al., 2015) and graph-based semi-supervised learning (Zhang and Duh, 2020) use surrogate models to search efficiently for optimal hyperparameters, avoiding inefficient grid search or manual tuning. Complementary approaches are successive halving (SHA; Jamieson and Talwalkar, 2016) and its massively parallel variant, asynchronous SHA (ASHA; Li et al., 2020b), which test multiple hyperparameter settings in parallel for a fixed number of training iterations, then discard the half of the settings with the worst validation set performance.

The SMAC3 library (Lindauer et al., 2022) implements several BO strategies, including a budget-limited variant for expensive deep learning tasks, and is integrated into *auto-sklearn* (Feurer et al., 2022) and *auto-pytorch* (Zimmer et al., 2021). However, with limited computational budgets, both BO and ASHA may fail to identify good settings (Liu and Wang, 2021). It is unclear whether these methods can be used to choose random initial weights or to order training samples, which also affect model performance (Dodge et al., 2020).

9.2 Hyperparameter Transfer

To minimize the number of trials needed to find optimal hyperparameter settings, one can transfer knowledge from other datasets or tasks—similar to how an ML engineer might select reasonable settings by hand. Transferring hyperparameters can be especially beneficial during expensive stages in the NLP pipeline, such as pre-training. Transfer neural processes (Wei et al., 2021) provide a way to transfer observations, parameters, and configurations from previous tasks using Bayesian optimization with a neural process as the surrogate model. This can lead to more accurate models with fewer trials than conventional BO approaches, but has yet to be tested for large NLP models. Finally, the cost of training can be reduced using μTransfer (Yang et al., 2021), which tunes a small model, then transfers the hyperparameters to a larger model.

9.3 Model Selection Considerations

While identifying an optimal model is crucial in deployment, it raises several challenges around reporting practices (Reimers and Gurevych, 2017; Agarwal et al., 2021) and hyperparameter tuning (Bouthillier and Varoquaux, 2020; Gundersen et al., 2022).[3] A first step towards improved comparability could be to fix the hyperparameter tuning budget (Dodge et al., 2019; Hoffmann et al., 2022), or consider the full search space (Bell et al., 2022).

10 Conclusion

This survey provides a broad overview of considerations for increasing efficiency in modern NLP models, identifying both immediate successes and remaining challenges. Most progress so far has been in model design, typically targeted at a specific computational budget and hardware paradigm. Key challenges include better understanding and modeling trade-offs between end-task performance and resource consumption, and the dependency between hardware choices and software implementations. Furthermore, we note that efficiency in NLP has many definitions and can be achieved in many different ways, but is also subject to various open challenges, and cannot be measured by a single metric. We outline several promising research directions aligned with overcoming these challenges, ranging from approaches that make better use of available data, strategies for reducing the cost of pre-training and fine-tuning large models, to prioritizing the importance of interactions between algorithms, software, and hardware.

Impressive advances in NLP enabled primarily by scaling computation have produced remarkable progress in a short span of time. However, in order to realize the full potential of this technology for a broader swath of society, we must reduce the amount of computation that is required to achieve these remarkable results. We hope that this survey can serve to accelerate advances in this important area of research with great potential for impact both within our field and for society as a whole.

[3]For example, when considering compute budget variation when comparing new model development to baselines.

Acknowledgments

This work was initiated at and benefited substantially from the Dagstuhl Seminar 22232: *Efficient and Equitable Natural Language Processing in the Age of Deep Learning*. We further thank Yuki Arase, Jonathan Frankle, Alexander Koller, Alexander Löser, Alexandra Sasha Luccioni, Haritz Puerto, Nils Reimers, Leonardo Riberio, Anna Rogers, Andreas Rücklé, Noah A. Smith, and Thomas Wolf for a fruitful discussion and helpful feedback at the seminar. M.T. and A.M. acknowledge the European Research Council (ERC StG DeepSPIN 758969), EU's Horizon Europe Research and Innovation Actions (UTTER, contract 101070631), and Fundação para a Ciência e Tecnologia through contract UIDB/50008/2020. L.D. acknowledges support of the Independent Research Fund Denmark under project 9131-00131B, Verif-AI, and the Novo Nordisk Foundation project ClinRead, NNF19-OC0059138. Finally, we also thank the TACL reviewers and action editor for helpful discussion and insightful feedback.

References

Chirag Agarwal, Daniel D'souza, and Sara Hooker. 2022. Estimating example difficulty using variance of gradients. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10368–10378. https://doi.org/10.1109/CVPR52688.2022.01012

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C. Courville, and Marc Bellemare. 2021. Deep reinforcement learning at the edge of the statistical precipice. In *Advances in Neural Information Processing Systems*, volume 34, pages 29304–29320. Curran Associates, Inc.

Armen Aghajanyan, Anchit Gupta, Akshat Shrivastava, Xilun Chen, Luke Zettlemoyer, and Sonal Gupta. 2021a. Muppet: Massive multi-task representations with pre-finetuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5799–5811, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.emnlp-main.468

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 2021b. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7319–7328, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.acl-long.568

Ameeta Agrawal, Suresh Singh, Lauren Schneider, and Michael Samuels. 2021. On the role of corpus ordering in language modeling. In *Proceedings of the Second Workshop on Simple and Efficient Natural Language Processing*, pages 142–154, Virtual. Association for Computational Linguistics.

Orevaoghene Ahia, Julia Kreutzer, and Sara Hooker. 2021. The low-resource double bind: An empirical study of pruning for low-resource machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3316–3333, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Nur Ahmed and Muntasir Wahed. 2020. The de-democratization of AI: Deep learning and the compute divide in artificial intelligence research. *arXiv preprint arXiv:2010.15581v1*.

Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. 2020. ETC: Encoding long and structured inputs in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 268–284, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.19

Ahmed Alajrami and Nikolaos Aletras. 2022. How does the pre-training objective affect what large language models learn about linguistic properties? In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 131–147, Dublin, Ireland. Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.acl-short.16

Uri Alon, Frank Xu, Junxian He, Sudipta Sengupta, Dan Roth, and Graham Neubig. 2022. Neuro-symbolic language modeling with automaton-augmented retrieval. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 468–485. PMLR.

Lasse F. Wolff Anthony, Benjamin Kanding, and Raghavendra Selvan. 2020. CarbonTracker: Tracking and predicting the carbon footprint of training deep learning models. In *Proceedings of the workshop on Challenges in Deploying and monitoring Machine Learning Systems, ICML*.

Vamsi Aribandi, Yi Tay, Tal Schuster, Jinfeng Rao, Huaixiu Steven Zheng, Sanket Vaibhav Mehta, Honglei Zhuang, Vinh Q. Tran, Dara Bahri, Jianmo Ni, Jai Gupta, Kai Hui, Sebastian Ruder, and Donald Metzler. 2022. ExT5: Towards extreme multi-task scaling for transfer learning. In *International Conference on Learning Representations*.

Jordan T. Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. 2020. Deep batch active learning by diverse, uncertain gradient lower bounds. In *International Conference on Learning Representations*.

Stephen Bach, Victor Sanh, Zheng Xin Yong, Albert Webson, Colin Raffel, Nihal V. Nayak, Abheesht Sharma, Taewoon Kim, M. Saiful Bari, Thibault Fevry, Zaid Alyafeai, Manan Dey, Andrea Santilli, Zhiqing Sun, Srulik Ben-david, Canwen Xu, Gunjan Chhablani, Han Wang, Jason Fries, Maged Al-shaibani, Shanya Sharma, Urmish Thakker, Khalid Almubarak, Xiangru Tang, Dragomir Radev, Mike Tian-Jian Jiang, and Alexander Rush. 2022. PromptSource: An integrated development environment and repository for natural language prompts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 93–104, Dublin, Ireland. Association for Computational Linguistics. `https://doi.org/10.18653/v1/2022.acl-demo.9`

Haoli Bai, Wei Zhang, Lu Hou, Lifeng Shang, Jin Jin, Xin Jiang, Qun Liu, Michael Lyu, and Irwin King. 2021. BinaryBERT: Pushing the limit of BERT quantization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4334–4348, Online. Association for Computational Linguistics. `https://doi.org/10.18653/v1/2021.acl-long.334`

Robert Baldock, Hartmut Maennel, and Behnam Neyshabur. 2021. Deep learning through the lens of example difficulty. In *Advances in Neural Information Processing Systems*, volume 34, pages 10876–10889. Curran Associates, Inc.

Ankur Bapna and Orhan Firat. 2019. Simple, scalable adaptation for neural machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1538–1548, Hong Kong, China. Association for Computational Linguistics.

Paul Barham, Aakanksha Chowdhery, Jeff Dean, Sanjay Ghemawat, Steven Hand, Daniel Hurt, Michael Isard, Hyeontaek Lim, Ruoming Pang, Sudip Roy, Brennan Saeta, Parker Schuh, Ryan Sepassi, Laurent Shafey, Chandu Thekkath, and Yonghui Wu. 2022. Pathways: Asynchronous distributed dataflow for ML. *Proceedings of Machine Learning and Systems*, 4:430–449.

Maximiliana Behnke and Kenneth Heafield. 2021. Pruning neural machine translation for speed using group lasso. In *Proceedings of the Sixth Conference on Machine Translation*, pages 1074–1086, Online. Association for Computational Linguistics.

Samuel Bell, Onno Kampman, Jesse Dodge, and Neil D. Lawrence. 2022. Modeling the machine learning multiverse. In *Advances in Neural Information Processing Systems*.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150v2*.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9, Dublin, Ireland. Association for Computational Linguistics. `https://doi.org/10.18653/v1/2022.acl-short.1`

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 41–48.

Aishwarya Bhandare, Vamsi Sripathi, Deepthi Karkada, Vivek Menon, Sun Choi, Kushal Datta, and Vikram Saletore. 2019. Efficient 8-bit quantization of transformer neural machine language translation model. In *Proceedings of the Joint Workshop on On-Device Machine Learning & Compact Deep Neural Network Representations, 36th International Conference on Machine Learning*.

Alexandra Birch, Andrew Finch, Hiroaki Hayashi, Kenneth Heafield, Marcin Junczys-Dowmunt, Ioannis Konstas, Xian Li, Graham Neubig, and Yusuke Oda, editors. 2020. *Proceedings of* *the Fourth Workshop on Neural Generation and Translation*. Association for Computational Linguistics, Online.

Yonatan Bitton, Michael Elhadad, Gabriel Stanovsky, and Roy Schwartz. 2021. Data efficient masked language modeling for vision and language. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3013–3028, Punta Cana, Dominican Republic. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.findings-emnlp.259

Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. 2020. What is the state of neural network pruning? *Proceedings of Machine Learning and Systems*, 2:129–146.

Zalán Bodó, Zsolt Minier, and Lehel Csató. 2011. Active learning with clustering. In *Active Learning and Experimental Design Workshop In conjunction with AISTATS 2010*, pages 127–139. JMLR Workshop and Conference Proceedings.

Nikolay Bogoychev, Roman Grundkiewicz, Alham Fikri Aji, Maximiliana Behnke, Kenneth Heafield, Sidharth Kashyap, Emmanouil-Ioannis Farsarakis, and Mateusz Chudyk. 2020. Edinburgh's submissions to the 2020 machine translation efficiency task. In *Proceedings of the Fourth Workshop on Neural Generation and Translation*, pages 218–224, Online. Association for Computational Linguistics.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego De Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack Rae, Erich Elsen, and Laurent Sifre. 2022. Improving language models by retrieving from trillions of tokens. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 2206–2240. PMLR.

Xavier Bouthillier and Gaël Varoquaux. 2020. Survey of machine-learning experimental methods at NeurIPS2019 and ICLR2020. Research report, Inria Saclay Ile de France.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Qingqing Cao, Aruna Balasubramanian, and Niranjan Balasubramanian. 2020. Towards accurate and reliable energy measurement of NLP models. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, pages 141–148, Online. Association for Computational Linguistics.

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28(1):41–75. https://doi.org/10.1023/A:1007379606734

Beidi Chen, Tri Dao, Kaizhao Liang, Jiaming Yang, Zhao Song, Atri Rudra, and Christopher Re. 2022. Pixelated butterfly: Simple and efficient sparse training for neural network models. In *International Conference on Learning Representations*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374v2*.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509v1*.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J. Colwell, and Adrian Weller. 2021. Rethinking attention with performers. In *International Conference on Learning Representations*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. PaLM: Scaling language modeling with pathways. arXiv:2204.02311v5.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.

Gonçalo M. Correia, Vlad Niculae, and André F. T. Martins. 2019. Adaptively sparse transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2174–2184, Hong Kong, China. Association for Computational Linguistics.

Corinna Cortes, Mehryar Mohri, Michael Riley, and Afshin Rostamizadeh. 2008. Sample selection bias correction theory. In *Algorithmic Learning Theory*, pages 38–53, Berlin, Heidelberg. Springer Berlin Heidelberg.

Raj Dabre, Raphael Rubino, and Atsushi Fujita. 2020. Balancing cost and benefit with tied-multi transformers. In *Proceedings of the Fourth Workshop on Neural Generation and Translation*, pages 24–34, Online. Association for Computational Linguistics.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy. Association for Computational Linguistics.

Tri Dao, Beidi Chen, Nimit S. Sohoni, Arjun Desai, Michael Poli, Jessica Grogan, Alexander Liu, Aniruddh Rao, Atri Rudra, and Christopher Ré. 2022a. Monarch: Expressive structured matrices for efficient and accurate training. In *International Conference on Machine Learning*, pages 4690–4721. PMLR.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Re. 2022b. FlashAttention: fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*.

Giannis Daras, Nikita Kitaev, Augustus Odena, and Alexandros G. Dimakis. 2020. SMYRF - Efficient attention using asymmetric clustering. In *Advances in Neural Information Processing Systems*, volume 33, pages 6476–6489. Curran Associates, Inc.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2019. Universal transformers. In *International Conference on Learning Representations*.

Mostafa Dehghani, Yi Tay, Anurag Arnab, Lucas Beyer, and Ashish Vaswani. 2022. The efficiency misnomer. In *International Conference on Learning Representations*.

Leon Derczynski. 2020. Power consumption variation over activation functions. *arXiv preprint arXiv:2006.07237v1*.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022a. GPT3.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems*.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 2022b. 8-bit optimizers via block-wise quantization. In *International Conference on Learning Representations*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A. Smith. 2019. Show your work: Improved reporting of experimental results. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2185–2194, Hong Kong, China. Association for Computational Linguistics. `https://doi.org/10.18653/v1/D19-1224`

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pre-trained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305v1*.

Jesse Dodge, Taylor Prewitt, Remi Tachet des Combes, Erika Odmark, Roy Schwartz, Emma Strubell, Alexandra Sasha Luccioni, Noah A. Smith, Nicole DeCario, and Will Buchanan. 2022. Measuring the carbon intensity of AI in cloud instances. In *2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22, pages 1877–1894, New York, NY, USA. Association for Computing Machinery. `https://doi.org/10.1145/3531146.3533234`

Xin Dong, Shangyu Chen, and Sinno Pan. 2017. Learning to prune deep neural networks via layer-wise optimal brain surgeon. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Daniel D'souza, Zach Nussbaum, Chirag Agarwal, and Sara Hooker. 2021. A tale of two long tails. *arXiv preprint arXiv:2107.13098v1*.

Nan Du, Yanping Huang, Andrew M. Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten P Bosma, Zongwei Zhou, Tao Wang, Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathleen Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc Le, Yonghui Wu, Zhifeng Chen, and Claire Cui. 2022. GLaM: Efficient scaling of language models with mixture-of-experts. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 5547–5569. PMLR.

Yann Dubois, Gautier Dagan, Dieuwke Hupkes, and Elia Bruni. 2020. Location attention for extrapolation to longer sequences. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 403–413, Online. Association for Computational Linguistics.

Liat Ein-Dor, Alon Halfon, Ariel Gera, Eyal Shnarch, Lena Dankin, Leshem Choshen, Marina Danilevsky, Ranit Aharonov, Yoav Katz, and Noam Slonim. 2020. Active learning for BERT: An empirical study. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7949–7962, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.638

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. 2020. Depth-adaptive transformer. In *International Conference on Learning Representations*.

Jeffrey L. Elman. 1993. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99. https://doi.org/10.1016/0010-0277(93)90058-4, PubMed: 8403835

Kawin Ethayarajh, Yejin Choi, and Swabha Swayamdipta. 2022. Understanding dataset difficulty with $\mathcal{V}$-usable information. In *International Conference on Machine Learning*, pages 5988–6008. PMLR.

Angela Fan, Edouard Grave, and Armand Joulin. 2020. Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations*.

William Fedus, Jeff Dean, and Barret Zoph. 2022a. A review of sparse expert models in deep learning. *arXiv preprint arXiv:2209.01667v1*.

William Fedus, Barret Zoph, and Noam Shazeer. 2022b. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, and Frank Hutter. 2022. Auto-Sklearn 2.0: Hands-free autoML via meta-learning. *Journal of Machine Learning Research*, 23(261):1–61.

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. 2015. Efficient and robust automated machine learning. *Advances in Neural Information Processing Systems*, 28.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. 2017. Deep Bayesian active learning with image data. In *International Conference on Machine Learning*, pages 1183–1192. PMLR.

Trevor Gale, Erich Elsen, and Sara Hooker. 2019. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574v1*.

Tao Ge, Si-Qing Chen, and Furu Wei. 2022. EdgeFormer: A parameter-efficient transformer for on-device seq2seq generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10786–10798, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé Iii, and Kate Crawford. 2021. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92. https://doi.org/10.1145/3458723

Daniel Gissin and Shai Shalev-Shwartz. 2019. Discriminative active learning. *arXiv preprint arXiv:1907.06347v1*.

Mitchell Gordon, Kevin Duh, and Nicholas Andrews. 2020. Compressing BERT: Studying the effects of weight pruning on transfer learning. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 143–155, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.repl4nlp-1.18

Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819. https://doi.org/10.1007/s11263-021-01453-z

Albert Gu, Karan Goel, Ankit Gupta, and Christopher Ré. 2022a. On the parameterization and initialization of diagonal state space models. In *Advances in Neural Information Processing Systems*.

Albert Gu, Karan Goel, and Christopher Re. 2022b. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor O. K. Li. 2018. Search engine guided non-parametric neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Odd Erik Gundersen, Kevin Coakley, and Christine Kirkpatrick. 2022. Sources of irreproducibility in machine learning: A review. *arXiv preprint arXiv:2204.07610v1*.

Demi Guo, Alexander Rush, and Yoon Kim. 2021. Parameter-efficient transfer learning with diff pruning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4884–4896, Online. Association for Computational Linguistics. `https://doi.org/10.18653/v1/2021.acl-long.378`

Ankit Gupta, Albert Gu, and Jonathan Berant. 2022. Diagonal state spaces are as effective as structured state spaces. In *Advances in Neural Information Processing Systems*.

Tae Jun Ham, Sung Jun Jung, Seonghak Kim, Young H. Oh, Yeonhong Park, Yoonho Song, Jung-Hun Park, Sanghee Lee, Kyoung Park, Jae W. Lee, and Deog-Kyoon Jeong. 2020. A^3: Accelerating attention mechanisms in neural networks with approximation. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 328–341.

Tae Jun Ham, Yejin Lee, Seong Hoon Seo, Soosung Kim, Hyunji Choi, Sung Jun Jung, and Jae W. Lee. 2021. ELSA: Hardware-software co-design for efficient, lightweight self-attention mechanism in neural networks. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, pages 692–705.

Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural networks. *Advances in Neural Information Processing Systems*, 28.

Michael Hassid, Hao Peng, Daniel Rotem, Jungo Kasai, Ivan Montero, Noah A. Smith, and Roy Schwartz. 2022. How much does attention actually attend? Questioning the importance of attention in pre-trained transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1403–1416, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jiaao He, Jidong Zhai, Tiago Antunes, Haojie Wang, Fuwen Luo, Shangfeng Shi, and Qin Li. 2022a. FasterMoE: Modeling and optimizing training of large-scale dynamic pre-trained models. In *Proceedings of the 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP '22, pages 120–134, New York, NY, USA. Association for Computing Machinery.

Junxian He, Graham Neubig, and Taylor Berg-Kirkpatrick. 2021. Efficient nearest neighbor language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5703–5714.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022b. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*.

Kaiming He, Ross Girshick, and Piotr Dollár. 2019. Rethinking ImageNet pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. `https://doi.org/10.1109/ICCV.2019.00502`

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. DeBERTaV3: Improving DeBERTa using electra-style pre-training with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations*.

Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. 2020. Towards the systematic reporting of the energy and carbon footprints of machine learning. *Journal of Machine Learning Research*, 21(248):1–43.

Daniel Hershcovich, Nicolas Webersinke, Mathias Kraus, Julia Bingler, and Markus Leippold. 2022. Towards climate awareness in NLP research. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2480–2494, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Marius Hessenthaler, Emma Strubell, Dirk Hovy, and Anne Lauscher. 2022. Bridging fairness and environmental sustainability in natural language processing. In *Proceedings of the 2022*

Conference on Empirical Methods in Natural Language Processing, pages 7817–7836, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Geoffrey Hinton. 2022. The forward-forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345v1*.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. In *NeurIPS Deep Learning and Representation Learning Workshop*.

Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. 2021. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack William Rae, and Laurent Sifre. 2022. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems*.

Sara Hooker. 2021. The hardware lottery. *Communications of the ACM*, 64:58–65. https://doi.org/10.1145/3467017

Sara Hooker, Nyalleng Moorosi, Gregory Clark, Samy Bengio, and Emily Denton. 2020. Characterising bias in compressed models. *arXiv preprint arXiv:2010.03058v1*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*.

Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861v1*.

C.-H. Hsu, W.-C. Feng, and Jeremy S. Archuleta. 2005. Towards efficient supercomputing: A quest for the right metric. In *19th IEEE International Parallel and Distributed Processing Symposium*, pages 8–pp. IEEE.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Zhiqi Huang, Lu Hou, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. 2021. GhostBERT: Generate more features with cheap operations for BERT. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6512–6523, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.acl-long.509

Itay Hubara, Yury Nahshan, Yair Hanani, Ron Banner, and Daniel Soudry. 2021. Accurate post training quantization with small calibration sets. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 4466–4475. PMLR.

Forrest Iandola, Albert Shaw, Ravi Krishna, and Kurt Keutzer. 2020. SqueezeBERT: What can computer vision teach NLP about efficient neural networks? In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, pages 124–135, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.sustainlp-1.17

Eugenia Iofinova, Alexandra Peste, Mark Kurtz, and Dan Alistarh. 2022. How well do sparse imagenet models transfer? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12266–12276. https://doi.org/10.1109/CVPR52688.2022.01195

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. 1991. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87. https://doi.org

/10.1162/neco.1991.3.1.79, PubMed: 31141872

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. 2021. Perceiver: General perception with iterative attention. In *International conference on machine learning*, pages 4651–4664. PMLR.

Kevin Jamieson and Ameet Talwalkar. 2016. Non-stochastic best arm identification and hyperparameter optimization. In *Artificial intelligence and statistics*, pages 240–248. PMLR.

William Stanley Jevons. 1866. *The Coal Question; An Inquiry Concerning the Progress of the Nation, and the Probable Exhaustion of Our Coal Mines*. Macmillan & Co. London.

Tianchu Ji, Shraddhan Jain, Michael Ferdman, Peter Milder, H. Andrew Schwartz, and Niranjan Balasubramanian. 2021. On the distribution, sparsity, and inference-time quantization of attention values in transformers. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4147–4157, Online. Association for Computational Linguistics.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.findings-emnlp.372

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361v1*.

Siddharth Karamcheti, Ranjay Krishna, Li Fei-Fei, and Christopher Manning. 2021. Mind your outliers! Investigating the negative impact of outliers on active learning for visual question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7265–7281, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.acl-long.564

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. Compacter: Efficient low-rank hypercomplex adapter layers. In *Advances in Neural Information Processing Systems*, volume 34.

Rabeeh Karimi Mahabadi, Luke Zettlemoyer, James Henderson, Lambert Mathias, Marzieh Saeidi, Veselin Stoyanov, and Majid Yazdani. 2022. Prompt-free and efficient few-shot learning with language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3638–3652, Dublin, Ireland. Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.acl-long.254

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR.

Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2021. Nearest neighbor machine translation. In *International Conference on Learning Representations*.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*.

Sehoon Kim, Amir Gholami, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. 2021. I-BERT: Integer-only BERT quantization. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5506–5518. PMLR.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics.

Young Jin Kim, Marcin Junczys-Dowmunt, Hany Hassan, Alham Fikri Aji, Kenneth Heafield, Roman Grundkiewicz, and Nikolay Bogoychev. 2019. From research to production and back: Ludicrously fast neural machine translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 280–288, Hong Kong. Association for Computational Linguistics.

Andreas Kirsch, Joost van Amersfoort, and Yarin Gal. 2019. BatchBALD: Efficient and diverse batch acquisition for deep Bayesian active learning. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *International Conference on Learning Representations*.

Jan-Christoph Klie, Richard Eckart de Castilho, and Iryna Gurevych. 2020. From zero to hero: Human-in-the-loop entity linking in low resource domains. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6982–6993, Online. Association for Computational Linguistics.

Julia Kreutzer, Isaac Caswell, Lisa Wang, Ahsan Wahab, Daan van Esch, Nasanbayar Ulzii-Orshikh, Allahsera Tapo, Nishant Subramani, Artem Sokolov, Claytone Sikasote, Monang Setyawan, Supheakmungkol Sarin, Sokhar Samb, Benoît Sagot, Clara Rivera, Annette Rios, Isabel Papadimitriou, Salomey Osei, Pedro Ortiz Suarez, Iroro Orife, Kelechi Ogueji, Andre Niyongabo Rubungo, Toan Q. Nguyen, Mathias Müller, André Müller, Shamsuddeen Hassan Muhammad, Nanda Muhammad, Ayanda Mnyakeni, Jamshidbek Mirzakhalov, Tapiwanashe Matangira, Colin Leong, Nze Lawson, Sneha Kudugunta, Yacine Jernite, Mathias Jenny, Orhan Firat, Bonaventure F. P. Dossou, Sakhile Dlamini, Nisansa de Silva, Sakine Çabuk Ballı, Stella Biderman, Alessia Battisti, Ahmed Baruwa, Ankur Bapna, Pallavi Baljekar, Israel Abebe Azime, Ayodele Awokoya, Duygu Ataman, Orevaoghene Ahia, Oghenefego Ahia, Sweta Agrawal, and Mofetoluwa Adeyemi. 2022. Quality at a glance: An audit of web-crawled multilingual datasets. *Transactions of the Association for Computational Linguistics*, 10:50–72. https://doi.org/10.1162/tacl_a_00447

M. Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc.

Andrey Kuzmin, Mart Van Baalen, Yuwei Ren, Markus Nagel, Jorn Peters, and Tijmen Blankevoort. 2022. FP8 quantization: The power of the exponent. In *Advances in Neural Information Processing Systems*.

Imad Lakim, Ebtesam Almazrouei, Ibrahim Abualhaol, Merouane Debbah, and Julien Launay. 2022. A holistic assessment of the carbon footprint of Noor, a very large Arabic language model. In *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*, pages 84–94, virtual+Dublin. Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.bigscience-1.8

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A lite BERT for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Ronan Le Bras, Swabha Swayamdipta, Chandra Bhagavatula, Rowan Zellers, Matthew Peters, Ashish Sabharwal, and Yejin Choi. 2020. Adversarial filters of dataset biases. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1078–1088. PMLR.

Yann LeCun, John Denker, and Sara Solla. 1989. Optimal brain damage. In *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann.

Ji-Ung Lee, Jan-Christoph Klie, and Iryna Gurevych. 2022a. Annotation curricula to implicitly train non-expert annotators. *Computational Linguistics*, 48(2):343–373. https://doi.org/10.1162/coli_a_00436

Ji-Ung Lee, Christian M. Meyer, and Iryna Gurevych. 2020. Empowering active learning to jointly optimize system and user demands. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4233–4247, Online. Association for Computational Linguistics.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2022b. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8424–8445, Dublin, Ireland. Association for Computational Linguistics.

James Lee-Thorp, Joshua Ainslie, Ilya Eckstein, and Santiago Ontanon. 2022. FNet: Mixing tokens with Fourier transforms. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4296–4313, Seattle, United States. Association for Computational Linguistics. `https://doi.org/10.18653/v1/2022.naacl-main.319`

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2021. {GS}hard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*.

Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. 2020. *Mining of Massive Data Sets*. Cambridge University Press.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

David D. Lewis and William A. Gale. 1994. A sequential algorithm for training text classifiers. In *SIGIR '94*, pages 3–12, London. Springer London.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics. `https://doi.org/10.18653/v1/2020.acl-main.703`

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020b. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Belinda Z. Li, Gabriel Stanovsky, and Luke Zettlemoyer. 2020a. Active learning for coreference resolution using discrete annotation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8320–8331, Online. Association for Computational Linguistics.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. 2018. Measuring the intrinsic dimension of objective landscapes. In *International Conference on Learning Representations*.

Huayang Li, Yixuan Su, Deng Cai, Yan Wang, and Lemao Liu. 2022a. A survey on retrieval-augmented text generation. *arXiv preprint arXiv:2202.01110v1*.

Liam Li, Kevin Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Jonathan Ben-tzur, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. 2020b. A system for massively parallel hyperparameter tuning. In *Third Conference on Systems and Machine Learning*.

Qin Li, Xiaofan Zhang, Jinjun Xiong, Wen-Mei Hwu, and Deming Chen. 2021. Efficient methods for mapping neural machine translator on FPGAs. *IEEE Transactions on Parallel and Distributed Systems*, 32(7):1866–1877. Conference Name: IEEE Transactions on Parallel and Distributed Systems. `https://doi.org/10.1109/TPDS.2020.3047371`

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual*

Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 4582–4597, Online. Association for Computational Linguistics.

Yuhong Li, Tianle Cai, Yi Zhang, Deming Chen, and Debadeepta Dey. 2022b. What makes convolutional models great on long sequence modeling? *arXiv preprint arXiv:2210.09298v1*.

Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joey Gonzalez. 2020c. Train big, then compress: Rethinking model size for efficient training and inference of transformers. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5958–5968. PMLR.

Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. 2022. SMAC3: A versatile Bayesian optimization package for hyperparameter optimization. *Journal of Machine Learning Research*, 23:54–1.

Haokun Liu, Derek Tam, Muqeeth Mohammed, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022a. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In *Advances in Neural Information Processing Systems*.

Ming Liu, Wray Buntine, and Gholamreza Haffari. 2018. Learning to actively learn neural machine translation. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 334–344, Brussels, Belgium. Association for Computational Linguistics.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9). https://doi.org/10.1145/3560815

Weijie Liu, Peng Zhou, Zhiruo Wang, Zhe Zhao, Haotang Deng, and Qi Ju. 2020. FastBERT: A self-distilling BERT with adaptive inference time. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6035–6044, Online. Association for Computational Linguistics.

Xiangyang Liu, Tianxiang Sun, Junliang He, Jiawen Wu, Lingling Wu, Xinyu Zhang, Hao Jiang, Zhao Cao, Xuanjing Huang, and Xipeng Qiu. 2022b. Towards efficient NLP: A standard evaluation and a strong baseline. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3288–3303, Seattle, United States. Association for Computational Linguistics.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021a. GPT understands, too. *arXiv preprint arXiv:2103.10385v1*.

Xueqing Liu and Chi Wang. 2021. An empirical study on hyperparameter optimization for fine-tuning pre-trained language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2286–2300, Online. Association for Computational Linguistics.

Zejian Liu, Gang Li, and Jian Cheng. 2021b. Hardware acceleration of fully quantized BERT for efficient natural language processing. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*.

Christos Louizos, Max Welling, and Diederik P. Kingma. 2018. Learning sparse neural networks through L_0 regularization. In *International Conference on Learning Representations*.

David Lowell, Zachary C. Lipton, and Byron C. Wallace. 2019. Practical obstacles to deploying active learning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 21–30, Hong Kong, China. Association for Computational Linguistics.

Siyuan Lu, Meiqi Wang, Shuang Liang, Jun Lin, and Zhongfeng Wang. 2020. Hardware accelerator for multi-head attention and position-wise feed-forward in the transformer. In *2020*

IEEE 33rd International System-on-Chip Conference (SOCC), pages 84–89. IEEE.

Sasha Luccioni, Victor Schmidt, Alexandre Lacoste, and Thomas Dandres. 2019. Quantifying the carbon emissions of machine learning. In *NeurIPS 2019 Workshop on Tackling Climate Change with Machine Learning*.

Xuezhe Ma, Chunting Zhou, Xiang Kong, Junxian He, Liangke Gui, Graham Neubig, Jonathan May, and Luke Zettlemoyer. 2023. Mega: Moving average equipped gated attention. In *The Eleventh International Conference on Learning Representations*.

Francesca Manes-Rossi, Adriana Tiron-Tudor, Giuseppe Nicolò, and Gianluca Zanellato. 2018. Ensuring more sustainable reporting in europe using non-financial disclosure—De facto and de jure evidence. *Sustainability*, 10(4):1162. `https://doi.org/10.3390/su10041162`

Katerina Margatina, Giorgos Vernikos, Loïc Barrault, and Nikolaos Aletras. 2021. Active learning by acquiring contrastive examples. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 650–663, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Pedro Martins, Zita Marinho, and Andre Martins. 2022a. Efficient machine translation domain adaptation. In *Proceedings of the 1st Workshop on Semiparametric Methods in NLP: Decoupling Logic from Knowledge*, pages 23–29, Dublin, Ireland and Online. Association for Computational Linguistics. `https://doi.org/10.18653/v1/2022.spanlp-1.3`

Pedro Henrique Martins, Zita Marinho, and Andre Martins. 2022b. ∞-former: Infinite memory transformer. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5468–5485, Dublin, Ireland. Association for Computational Linguistics.

Pedro Henrique Martins, Zita Marinho, and André F. T. Martins. 2022c. Chunk-based nearest neighbor machine translation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4228–4245, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Harsh Mehta, Ankit Gupta, Ashok Cutkosky, and Behnam Neyshabur. 2023. Long range language modeling via gated state spaces. In *The Eleventh International Conference on Learning Representations*.

Yuxian Meng, Xiaoya Li, Xiayu Zheng, Fei Wu, Xiaofei Sun, Tianwei Zhang, and Jiwei Li. 2022. Fast nearest neighbor machine translation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 555–565, Dublin, Ireland. Association for Computational Linguistics. `https://doi.org/10.18653/v1/2022.findings-acl.47`

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, volume 32, pages 14014–14024. Curran Associates, Inc.

Swaroop Mishra and Bhavdeep Singh Sachdeva. 2020. Do we need to create big datasets to learn a task? In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, pages 169–173, Online. Association for Computational Linguistics. `https://doi.org/10.18653/v1/2020.sustainlp-1.23`

Alireza Mohammadshahi, Vassilina Nikoulina, Alexandre Berard, Caroline Brun, James Henderson, and Laurent Besacier. 2022. What do compressed multilingual machine translation models forget? In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4308–4329, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Nafise Moosavi, Quentin Delfosse, Kristian Kersting, and Iryna Gurevych. 2022. Adaptable adapters. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3742–3753, Seattle, United States. Association for Computational Linguistics. `https://doi.org/10.18653/v1/2022.naacl-main.274`

Hesham Mostafa and Xin Wang. 2019. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In *Proceedings of the 36th International Conference on Machine Learning*, pages 4646–4655. PMLR.

Basil Mustafa, Carlos Riquelme Ruiz, Joan Puigcerver, Rodolphe Jenatton, and Neil Houlsby. 2022. Multimodal contrastive learning with LIMoE: The language-image mixture of experts. In *Advances in Neural Information Processing Systems*.

Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. 2020. What is being transferred in transfer learning? In *Advances in Neural Information Processing Systems*, volume 33, pages 512–523. Curran Associates, Inc.

Badreddine Noune, Philip Jones, Daniel Justus, Dominic Masters, and Carlo Luschi. 2022. 8-bit numerical formats for deep neural networks. *arXiv preprint arXiv:2206.02915v1*.

Kelechi Ogueji, Orevaoghene Ahia, Gbemileke Onilude, Sebastian Gehrmann, Sara Hooker, and Julia Kreutzer. 2022. Intriguing properties of compression on multilingual models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9092–9110, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Vilfredo Pareto. 1896. *Cours d'Économie Politique professé à l'Université de Lausanne*, volume 1. F. Rouge.

David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. 2021. Carbon emissions and large neural network training. *arXiv preprint arXiv: 2104.10350v3*.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah Smith, and Lingpeng Kong. 2020. Random feature attention. In *International Conference on Learning Representations*.

Ben Peters and André F. T. Martins. 2021. Smoothing and shrinking the sparse seq2seq search space. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2642–2654, Online. Association for Computational Linguistics. `https://doi.org/10.18653/v1/2021.naacl-main.210`

Ben Peters, Vlad Niculae, and André F. T. Martins. 2019. Sparse sequence-to-sequence models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1504–1519, Florence, Italy. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics. `https://doi.org/10.18653/v1/N18-1202`

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics. `https://doi.org/10.18653/v1/D19-1250`

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020. AdapterHub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 46–54, Online. Association for Computational Linguistics. `https://doi.org/10.18653/v1/2020.emnlp-demos.7`

Edoardo M. Ponti, Alessandro Sordoni, and Siva Reddy. 2022. Combining modular skills in multitask learning. *arXiv preprint arXiv: 2202.13914v1*.

Gabriele Prato, Ella Charlaix, and Mehdi Rezagholizadeh. 2020. Fully quantized transformer for machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1–14, Online. Association for Computational Linguistics.

https://doi.org/10.18653/v1/2020.findings-emnlp.1

Ofir Press, Noah Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*.

Ofir Press, Noah A. Smith, and Mike Lewis. 2021. Shortformer: Better language modeling using shorter inputs. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5493–5505, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.acl-long.427

Lucas Høyberg Puvis de Chavannes, Mads Guldborg Kjeldgaard Kongsbak, Timmie Rantzau, and Leon Derczynski. 2021. Hyperparameter power impact in transformer language model training. In *Proceedings of the Second Workshop on Simple and Efficient Natural Language Processing*, pages 96–118, Virtual. Association for Computational Linguistics.

Zheng Qu, Liu Liu, Fengbin Tu, Zhaodong Chen, Yufei Ding, and Yuan Xie. 2022. DOTA: Detect and omit weak attentions for scalable transformer acceleration. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS 2022, pages 14–26, New York, NY, USA. Association for Computing Machinery.

Jerry Quinn and Miguel Ballesteros. 2018. Pieces of eight: 8-bit neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pages 114–120, New Orleans - Louisiana. Association for Computational Linguistics. https://doi.org/10.18653/v1/N18-3014

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444v2*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, H. Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant M. Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew J. Johnson, Blake A. Hechtman, Laura Weidinger, Iason Gabriel, William S. Isaac, Edward Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446v2*.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. 2020. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and

Yuxiong He. 2022. DeepSpeed-MoE: Advancing mixture-of-experts inference and training to power next-generation AI scale. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 18332–18346. PMLR.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2017. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Machel Reid, Edison Marrese-Taylor, and Yutaka Matsuo. 2021. Subformer: Exploring weight sharing for parameter efficiency in generative transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4081–4090. Association for Computational Linguistics, Punta Cana, Dominican Republic. `https://doi.org/10.18653/v1/2021.findings-emnlp.344`

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 338–348, Copenhagen, Denmark. Association for Computational Linguistics. `https://doi.org/10.18653/v1/D17-1035`

Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. 2021a. ZeRO-Offload: Democratizing billion-scale model training. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 551–564. USENIX Association.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B. Gupta, Xiaojiang Chen, and Xin Wang. 2021b. A survey of deep active learning. *ACM Computing Surveys*, 54(9). `https://doi.org/10.1145/3472291`

Adithya Renduchintala, Denise Diaz, Kenneth Heafield, Xian Li, and Mona Diab. 2021. Gender bias amplification during speed-quality optimization in neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 99–109, Online. Association for Computational Linguistics. `https://doi.org/10.18653/v1/2021.acl-short.15`

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2021. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68. `https://doi.org/10.1162/tacl_a_00353`

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. 2021. AdapterDrop: On the efficiency of adapters in transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7930–7946, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics. `https://doi.org/10.18653/v1/2021.emnlp-main.626`

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098v1*.

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. 2023. On the effect of dropping layers of pre-trained transformer models. *Computer Speech & Language*, 77:101429. `https://doi.org/10.1016/j.csl.2022.101429`

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. In *NeurIPS EMC^2 Workshop*.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M. Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and

Alexander M. Rush. 2022. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*.

Victor Sanh, Thomas Wolf, and Alexander Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning. In *Advances in Neural Information Processing Systems*, volume 33, pages 20378–20389. Curran Associates, Inc.

Chinnadhurai Sankar, Sujith Ravi, and Zornitsa Kozareva. 2021. ProFormer: Towards on-device LSH projection based transformers. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2823–2828, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.eacl-main.246

Timo Schick and Hinrich Schütze. 2021. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.naacl-main.185

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2020a. Green AI. *Communications of the ACM (CACM)*, 63(12):54–63. https://doi.org/10.1145/3381831

Roy Schwartz, Gabriel Stanovsky, Swabha Swayamdipta, Jesse Dodge, and Noah A. Smith. 2020b. The right tool for the job: Matching model and instance complexities. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6640–6651, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.593

Ozan Sener and Silvio Savarese. 2018. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*.

Burr Settles. 2012. *Active Learning*, volume 18 of *Synthesis Lectures on Artificial Intelligence and Machine Learning*. Morgan & Claypool.

Burr Settles, Mark Craven, and Lewis Friedland. 2008. Active learning with real annotation costs. In *Proceedings of the NIPS workshop on cost-sensitive learning (Vol. 1)*.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana. Association for Computational Linguistics. https://doi.org/10.18653/v1/N18-2074

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. 2020. Q-BERT: Hessian based ultra low precision quantization of BERT. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(5):8815–8821. Number: 5. https://doi.org/10.1609/aaai.v34i05.6409

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.346

Shoaib Ahmed Siddiqui, Nitarshan Rajkumar, Tegan Maharaj, David Krueger, and Sara Hooker. 2021. Metadata archaeology: Unearthing data subsets by leveraging training dynamics. *arXiv preprint arXiv:2209.10015v1*.

Andrew Silva, Pradyumna Tambwekar, and Matthew Gombolay. 2021. Towards a comprehensive understanding and accurate evaluation of societal biases in pre-trained transformers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2383–2389, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.naacl-main.189

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. 2012. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Samuel Stanton, Pavel Izmailov, Polina Kirichenko, Alexander A. Alemi, and Andrew G. Wilson. 2021. Does knowledge distillation really work? In *Advances in Neural Information Processing Systems*, volume 34, pages 6906–6919. Curran Associates, Inc.

Pierre Stock, Angela Fan, Benjamin Graham, Edouard Grave, Rémi Gribonval, Herve Jegou, and Armand Joulin. 2021. Training with quantization noise for extreme model compression. In *International Conference on Learning Representations*.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1355

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. MobileBERT: A compact task-agnostic BERT for resource-limited devices. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2158–2170, Online. Association for Computational Linguistics.

Yi-Lin Sung, Varun Nair, and Colin A. Raffel. 2021. Training neural networks with fixed sparse masks. In *Advances in Neural Information Processing Systems*, volume 34, pages 24193–24205. Curran Associates, Inc.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. 2020. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9275–9293, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.746

Thierry Tambe, Coleman Hooper, Lillian Pentecost, Tianyu Jia, En-Yu Yang, Marco Donato, Victor Sanh, Paul Whatmough, Alexander M. Rush, David Brooks, and Gu-Yeon Wei. 2021. EdgeBERT: Sentence-level energy optimizations for latency-aware multi-task NLP inference. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '21, pages 830–844, New York, NY, USA. Association for Computing Machinery. https://doi.org/10.1145/3466752.3480095

Min Tang, Xiaoqiang Luo, and Salim Roukos. 2002. Active learning for statistical natural language parsing. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 120–127, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. 2021. Long range arena : A benchmark for efficient transformers. In *International Conference on Learning Representations*.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2022. Efficient transformers: A survey. *ACM Computing Surveys*. https://doi.org/10.1145/3530811

Yi Tay, Shuohang Wang, Anh Tuan Luu, Jie Fu, Minh C. Phan, Xingdi Yuan, Jinfeng Rao, Siu Cheung Hui, and Aston Zhang. 2019. Simple and effective curriculum pointer-generator networks for reading comprehension over long narratives. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4922–4931, Florence, Italy. Association for Computational Linguistics.

Kale-ab Tessera, Sara Hooker, and Benjamin Rosman. 2021. Keep the gradients flowing: Using gradient flow to study sparse network optimization. *arXiv preprint arXiv:2102.01670v2*.

Neil C. Thompson, Kristjan Greenewald, Keeheon Lee, and Gabriel F. Manso. 2020. The computational limits of deep learning. *arXiv preprint arXiv:2007.05558v1*.

Marcos Treviso, António Góis, Patrick Fernandes, Erick Fonseca, and Andre Martins. 2022. Predicting attention sparsity in transformers. In *Proceedings of the Sixth Workshop on Structured Prediction for NLP*, pages 67–81, Dublin, Ireland. Association for Computational Linguistics.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. 2022. DyLoRA: Parameter efficient tuning of pre-trained models using dynamic search-free low rank adaptation. In *2nd Workshop on Efficiennt Natural Language and Speech Processing, (NeurIPS workshops)*, pages 1–6.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1580

Yu Wan, Baosong Yang, Derek F. Wong, Yikai Zhou, Lidia S. Chao, Haibo Zhang, and Boxing Chen. 2020. Self-paced learning for neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1074–1080, Online. Association for Computational Linguistics.

Hanrui Wang, Zhanghao Wu, Zhijian Liu, Han Cai, Ligeng Zhu, Chuang Gan, and Song Han. 2020a. HAT: Hardware-aware transformers for efficient natural language processing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7675–7688, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.686

Hanrui Wang, Zhekai Zhang, and Song Han. 2021a. SpAtten: Efficient sparse attention architecture with cascade token and head pruning. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 97–110. https://doi.org/10.1109/HPCA51647.2021.00018

Serena Wang, Harikrishna Narasimhan, Yichen Zhou, Sara Hooker, Michal Lukasik, and Aditya Krishna Menon. 2022a. Robust distillation for worst-class performance. *arXiv preprint arXiv:2206.06479v1*.

Shuhe Wang, Jiwei Li, Yuxian Meng, Rongbin Ouyang, Guoyin Wang, Xiaoya Li, Tianwei Zhang, and Shi Zong. 2021b. Faster nearest neighbor machine translation. *arXiv preprint arXiv:2112.08152v1*.

Yaqing Wang, Sahaj Agarwal, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao. 2022b. AdaMix: Mixture-of-adaptations for parameter-efficient model tuning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5744–5760, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Ziheng Wang, Jeremy Wohlwend, and Tao Lei. 2020b. Structured pruning of large language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6151–6162, Online. Association for Computational Linguistics.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022a. Fine-tuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022b. Emergent abilities of large language models. *Transactions on Machine Learning Research*. Survey Certification.

Ying Wei, Peilin Zhao, and Junzhou Huang. 2021. Meta-learning hyperparameter performance prediction with neural processes. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 11058–11067. PMLR.

Alexander Wettig, Tianyu Gao, Zexuan Zhong, and Danqi Chen. 2022. Should you mask 15% in masked language modeling? *arXiv preprint arXiv:2202.08005v1*.

Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga, Jinshi Huang, Charles Bai, Michael Gschwind, Anurag Gupta, Myle Ott, Anastasia Melnikov, Salvatore Candido, David Brooks, Geeta Chauhan, Benjamin Lee, Hsien-Hsin Lee, Bugra Akyildiz, Maximilian Balandat, Joe Spisak, Ravi Jain, Mike Rabbat, and Kim Hazelwood. 2022a. Sustainable AI: Environmental implications, challenges and opportunities. In *Proceedings of Machine Learning and Systems*, volume 4, pages 795–813.

Xiaoxia Wu, Zhewei Yao, Minjia Zhang, Conglong Li, and Yuxiong He. 2022b. Extreme compression for pre-trained transformers made simple and efficient. In *Advances in Neural Information Processing Systems*.

Zhanghao Wu, Zhijian Liu, Ji Lin, Yujun Lin, and Song Han. 2020. Lite transformer with long-short range attention. In *International Conference on Learning Representations*.

Mengzhou Xia, Zexuan Zhong, and Danqi Chen. 2022. Structured pruning learns compact and accurate models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1513–1528, Dublin, Ireland. Association for Computational Linguistics.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. DeeBERT: Dynamic early exiting for accelerating BERT inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2246–2251, Online. Association for Computational Linguistics. `https://doi.org/10.18653/v1/2020.acl-main.204`

Benfeng Xu, Licheng Zhang, Zhendong Mao, Quan Wang, Hongtao Xie, and Yongdong Zhang. 2020. Curriculum learning for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6095–6104, Online. Association for Computational Linguistics.

Canwen Xu and Julian McAuley. 2023. A survey on dynamic neural networks for natural language processing. In *Findings of EACL*.

Canwen Xu, Wangchunshu Zhou, Tao Ge, Ke Xu, Julian McAuley, and Furu Wei. 2021. Beyond preserved accuracy: Evaluating loyalty and robustness of BERT compression. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10653–10659, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Guangxuan Xu and Qingyuan Hu. 2022. Can Model compression improve NLP fairness. *arXiv preprint arXiv:2201.08542v1*.

Ge Yang, Edward Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. 2021. Tuning large neural networks via zero-shot hyperparameter transfer. In *Advances in Neural Information Processing Systems*, volume 34, pages 17084–17097. Curran Associates, Inc.

Dani Yogatama, Cyprien de Masson d'Autume, and Lingpeng Kong. 2021. Adaptive semi-parametric language models. *Transactions of the Association for Computational Linguistics*, 9:362–373. `https://doi.org/10.1162/tacl_a_00371`

Michelle Yuan, Hsuan-Tien Lin, and Jordan Boyd-Graber. 2020. Cold-start active learning through self-supervised language modeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7935–7948, Online. Association for Computational Linguistics.

Michelle Yuan, Patrick Xia, Chandler May, Benjamin Van Durme, and Jordan Boyd-Graber. 2022. Adapting coreference resolution models through active learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7533–7549, Dublin, Ireland. Association for Computational Linguistics. `https://doi.org/10.18653/v1/2022.acl-long.519`

A. H. Zadeh, I. Edo, O. M. Awad, and A. Moshovos. 2020. GOBO: Quantizing attention-based NLP models for low latency and energy efficient inference. In *2020 53rd Annual IEEE/ACM International Symposium* *on Microarchitecture (MICRO)*, pages 811–824. https://doi.org/10.1109/MICRO50266.2020.00071

Ali Hadi Zadeh, Mostafa Mahmoud, Ameer Abdelhadi, and Andreas Moshovos. 2022. Mokey: Enabling narrow fixed-point inference for out-of-the-box floating-point transformer models. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ISCA '22, pages 888–901, New York, NY, USA. Association for Computing Machinery. https://doi.org/10.1145/3470496.3527438

Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. 2019. Q8BERT: Quantized 8bit BERT. In *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS Edition (EMC^2-NIPS)*, pages 36–39. https://doi.org/10.1109/EMC2-NIPS53020.2019.00016

Ofir Zafrir, Ariel Larey, Guy Boudoukh, Haihao Shen, and Moshe Wasserblat. 2021. Prune once for all: Sparse pre-trained language models. *arXiv preprint arXiv:2111.05754v1*.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems*, volume 33, pages 17283–17297. Curran Associates, Inc.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 93–104, Brussels, Belgium. Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1009

Shuangfei Zhai, Walter Talbott, Nitish Srivastava, Chen Huang, Hanlin Goh, Ruixiang Zhang, and Josh Susskind. 2021. An attention free transformer. *arXiv preprint arXiv:2105.14103v1*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. OPT: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068v4*.

Wei Zhang, Lu Hou, Yichun Yin, Lifeng Shang, Xiao Chen, Xin Jiang, and Qun Liu. 2020. TernaryBERT: Distillation-aware ultra-low bit BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 509–521, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.37

Xuan Zhang and Kevin Duh. 2020. Reproducible and efficient benchmarks for hyperparameter optimization of neural machine translation systems. *Transactions of the Association for Computational Linguistics*, 8393–408.

Xuan Zhang, Pamela Shapiro, Gaurav Kumar, Paul McNamee, Marine Carpuat, and Kevin Duh. 2019. Curriculum learning for domain adaptation in neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1903–1915, Minneapolis, Minnesota. Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1189

Mingjun Zhao, Haijiang Wu, Di Niu, and Xiaoli Wang. 2020. Reinforced curriculum learning on pre-trained neural machine translation models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9652–9659.

Yikai Zhou, Baosong Yang, Derek F. Wong, Yu Wan, and Lidia S. Chao. 2020. Uncertainty-aware curriculum learning for neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6934–6944, Online. Association for Computational Linguistics.

Qingqing Zhu, Xiuying Chen, Pengfei Wu, JunFei Liu, and Dongyan Zhao. 2021. Combining curriculum learning and knowledge distillation for dialogue generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1284–1295, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yichen Zhu, Ning Liu, Zhiyuan Xu, Xin Liu, Weibin Meng, Louis Wang, Zhicai Ou, and Jian Tang. 2022. Teach less, learn more: On the undistillable classes in knowledge distillation. In *Advances in Neural Information Processing Systems*.

Lucas Zimmer, Marius Lindauer, and Frank Hutter. 2021. Auto-PyTorch: Multi-fidelity metalearning for efficient and robust autoDL. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(9):3079–3090. https://doi.org/10.1109/TPAMI.2021.3067763, PubMed: 33750687

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. 2022. Designing effective sparse expert models. In *2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 1044–1044. IEEE Computer Society. https://doi.org/10.1109/IPDPSW55747.2022.00171

Chapter 9

Empowering Active Learning to Jointly Optimize System and User Demands

Empowering Active Learning to Jointly Optimize System and User Demands

Ji-Ung Lee **Christian M. Meyer** **Iryna Gurevych**
{lastname}@ukp.informatik.tu-darmstadt.de
Ubiquitous Knowledge Processing (UKP) Lab
Computer Science Department
Technische Universität Darmstadt, Germany
https://www.ukp.tu-darmstadt.de

Abstract

Existing approaches to active learning maximize the system performance by sampling unlabeled instances for annotation that yield the most efficient training. However, when active learning is integrated with an end-user application, this can lead to frustration for participating users, as they spend time labeling instances that they would not otherwise be interested in reading. In this paper, we propose a new active learning approach that jointly optimizes the seemingly counteracting objectives of the active learning system (training efficiently) and the user (receiving useful instances). We study our approach in an educational application, which particularly benefits from this technique as the system needs to rapidly learn to predict the appropriateness of an exercise to a particular user, while the users should receive only exercises that match their skills. We evaluate multiple learning strategies and user types with data from real users and find that our joint approach better satisfies both objectives when alternative methods lead to many unsuitable exercises for end users.[1]

1 Introduction

State-of-the-art machine learning approaches require huge amounts of training data. But for many NLP applications, there is little to no training data available. *Interactive NLP systems* are a viable solution to alleviate the cost of creating large training datasets before a new application can be used. Such systems start with no or few labeled instances and acquire additional training data based on user feedback for their predictions. *Active learning* (Settles, 2012) is a frequently used technique to quickly maximize the prediction performance, as the system acquires user feedback in each iteration for those instances that likely yield the highest performance improvement (e.g., because the system is yet uncertain about them). Active learning has been shown to reduce the amount of user feedback required while improving system performance for interactive NLP systems (P.V.S and Meyer, 2017; Gao et al., 2018) and to reduce the annotation costs in crowdsourcing scenarios (Fang et al., 2014). However, outside the typical annotation setup, it can be boring or frustrating for users to provide feedback on ill-predicted instances that hardly solve their needs. Consider a newly launched web application for learning a foreign language, which aims at suggesting exercises that match the user's proficiency according to Vygotsky's *Zone of proximal development* (Vygotsky, 1978). The underlying machine learning system starts without any data, but employs active learning to select an exercise the system cannot confidently predict. Then, it adjusts its model interactively based on the user's feedback. While the system is still uncertain, the users often receive inappropriate (e.g., too hard or too easy) exercises. Thus, they get the impression that the system does not work properly, which is especially harmful during the inception phase of an application, as the community opinion largely defines its success.

In this paper, we distinguish the *system objective* of maximizing the prediction performance with minimal labeled instances and the *user objective* of providing useful instances for the user's current needs. For the first time, we propose an active learning approach that jointly optimizes these seemingly counteracting objectives and thus trades off the demands of system and user.

The users of educational applications can particularly benefit from this, as they can learn most if they receive appropriate learning material while the underlying system requires considerable training to reach acceptable performance. We employ our

[1] Our code and simulated learner models are available on Github: https://github.com/UKPLab/acl2020-empowering-active-learning

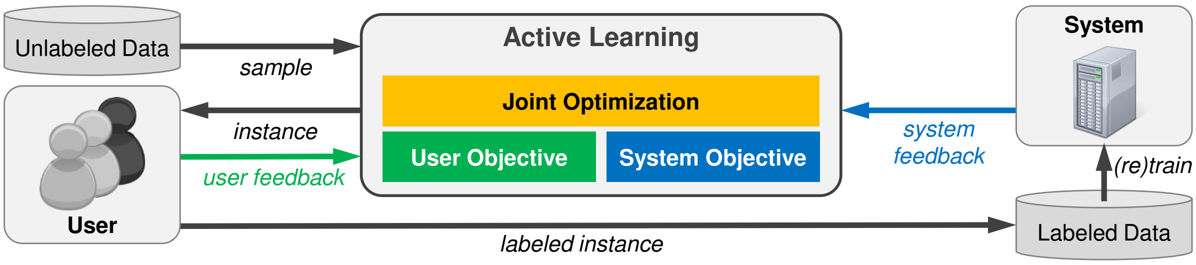


Figure 1: Overview of our interactive approach. We go beyond previous work on optimizing the system objective (blue) by modeling the user objective (green) and jointly optimizing these seemingly counteracting goals (gold).

new approach in a language learning platform for C-tests (i.e., cloze tests, in which the second half of every second word is replaced by a gap). Our system successfully learns how to predict the difficulty of a C-test gap (system objective) and how to provide a C-test that is neither too easy for the current user, which would cause boredom, nor too hard, which would create frustration (user objective). Predicting the difficulty of an exercise and correspondingly selecting exercises that match a user's proficiency are important steps towards self-directed language learning and massive open online courses (MOOCs) on language learning. Though we focus on this educational use case in this paper, our approach may also yield new insights for other problems that suffer from seemingly counteracting system and user objectives, for example, interactively trained recommender systems for books, movies, or restaurants.

2 Related Work

Active learning. Active learning aims to reduce the amount of training data by intelligently sampling instances that benefit the model most (Settles, 2012). A distinct characteristic of active learning is that labels for sampled instances are unknown and provided by an oracle after sampling. Various works investigate the use of active learning for crowdsourcing, where the oracles (i.e., the crowdworkers) may provide noisy labels (Snow et al., 2008; Laws et al., 2011). Within the educational domain, active learning research is scarce.[2] One example is the work by Rastogi et al. (2018), who propose a threshold-based sampling strategy utilizing the prediction probability and achieve a considerable speed-up without any significant performance drop. Hastings et al. (2018) find that active learning can be used to efficiently train a system for providing feedback on student essays using teachers as oracles. Horbach and Palmer (2016) report mixed results for employing active learning in short-answer grading. While all of these works focus on improvements of the proposed system, users only benefit after training. In contrast, our work explicitly models the user objective, such that users already benefit while labeling training instances.

Adaptive learning. Many systems provide user adaptation, and research has shifted from predefined sets of rules for adaptation to data-driven approaches. Several works investigate adaptive methods to provide exercises which are neither too hard nor too boring. For instance, Missura and Gärtner (2011) model learning in a game-theoretic sense where the goal is to adjust the difficulty to neither being too easy nor too hard. Other works investigate adaptation in the context of testing (Zheng and Chang, 2015; Wang et al., 2016; Chaimongkol et al., 2016) and propose methods for an adaptive selection of appropriate tests for better assessing a student's proficiency. In a large survey, Truong (2016) discusses how to integrate different learning styles, modeling categorical student behavior, into an adaptive learning environment and emphasizes the need for more sophisticated methods.

Despite much research in adaptive and active learning, none of the previous works consider jointly modeling and optimizing both the system and user objectives which may retain a user's motivation and keep them from leaving the platform due to boredom or frustration.

3 Approach

Figure 1 shows our proposed interactive learning setup. The *active learning* component iteratively samples instances from a pool of unlabeled data and asks the user for a label that can be used to

[2]Note, that in education, active learning often refers to a teaching paradigm which is unrelated to active learning in machine learning.

train the machine learning system. Previous work on active learning focused on optimizing the *system objective* (blue). That is, only the system provides feedback to the active learning component (e.g., how certain it is about the predicted label of an instance). In our work, we first model the *user objective* (green) and propose sampling strategies that maximize the user satisfaction based on the user's feedback (e.g., the user's label for an instance). Finally, we study our novel *joint optimization* strategies (gold) that trade off the demands of the system and the users. Whereas we distinguish between the user's feedback (exercise-level) and labeled instances (gap-level) in our work, our proposed approach can easily be adapted to more specific cases where the (implicit) user feedback and the provided label are the same.[3]

In the remainder of this section, we introduce *sampling strategies* that select which instance should be presented to the user next. We use the following notation: Let $\mathcal{X}$ be the pool of unlabeled instances. In every iteration of the application (e.g., when a user requests a new exercise), the sampling strategy $s(v)$ returns an instance $x \in \mathcal{X}$ for user v. The user then provides a label y for instance x, potentially with additional feedback on the user's satisfaction. The active learning component finally removes x from its pool $\mathcal{X}$ and adds (x, y) to the set of labeled instances, before the system is retrained with the increased labeled training set.

The simplest sampling strategy that we use as a baseline is *random sampling* $s_{\text{rand}}(v)$, which selects an $x \in \mathcal{X}$ uniformly at random, regardless of the user. In the following subsections, we discuss more advanced strategies that optimize the system or user objective as well as our new joint optimization strategies.

3.1 System optimization

To optimize the system objective, we consider *uncertainty sampling* (Lewis and Gale, 1994). Uncertainty sampling assumes that instances for which the model is least certain during prediction provide the most information for the model once their labels are known. The sampled instance is thus

$$s_{\text{unc}}(v) = \arg\max_{x \in \mathcal{X}} U(x) \tag{1}$$

where $U\colon x \mapsto [0,1]$ returns the uncertainty of predicting a label for instance x. Like random sampling, $s_{\text{unc}}(v)$ is independent of the current user v. A model's uncertainty can be measured in multiple different ways, for example, by the prediction probability of the predicted label (Lewis and Gale, 1994), as the difference in probabilities between the first and second most probable labels (Scheffer et al., 2001), and based on the Shannon entropy (Shannon, 1948) that considers all possible labels (Settles and Craven, 2008). We instantiate U for our educational application in section 4.

3.2 User optimization

The objective of users is to receive instances that meet their demands. We therefore define a new *user-oriented sampling* strategy as

$$s_{\text{usr}}(v) = \arg\max_{x \in \mathcal{X}} A(x, v) \tag{2}$$

where $A\colon (x, v) \mapsto [0,1]$ returns the degree of appropriateness of instance x for the user v. In our educational application, we consider an exercise appropriate if it is neither too easy nor too difficult, as this maximizes the user's learning gain. To quantify A, we measure the error between the predicted label $f(x)$ and the user's demand $\phi(v)$ as

$$A(x, v) = 1 - \text{err}[f(x), \phi(v)] \tag{3}$$

with an error function $\text{err} \in [0, 1]$ (cf., section 4).

3.3 Joint optimization

We propose two novel strategies to jointly optimize the user and system objectives.

Combined sampling. Our first strategy

$$s_{\text{comb}}(v) = \arg\max_{x \in \mathcal{X}} U(x)\, A(x, v) \tag{4}$$

combines uncertainty sampling and user-oriented sampling by preferring appropriate instances for user v (as in s_{usr}), but among them returns the one the system is most uncertain about (as in s_{unc}).

Trade-off sampling. For our second strategy, we aggregate both objectives into a single function

$$s_{\text{tos}}(v) = \arg\max_{x \in \mathcal{X}} \left\{ (1 - \lambda)\; A(x, v) + \lambda\, U(x) \right\} \tag{5}$$

which is the weighted sum of user-oriented and uncertainty sampling. The weight parameter $\lambda \in [0, 1]$ can be used to adjust the learning towards the system objective or the user objective.

[3] Note, that from a single answer which is either correct or wrong, we cannot deduce a fine-grained gap label. To obtain these in a real-world setting, one either may assume querying groups of users or asking them for an explicit label.

4 Instantiation

We consider our jointly optimized active learning particularly beneficial for educational applications, since (1) the users of such a system may fail to achieve their learning goals with inappropriate exercises. Additionally, (2) it is difficult to acquire large difficulty-annotated datasets for training, as actual users are required for producing realistic training data and existing learner datasets can hardly be shared due to privacy concerns. We therefore instantiate our approach for a language learning platform that predicts the difficulty of exercises and learns to provide appropriate (neither too easy nor too hard) exercises to its users.

C-tests. For our experiments, we use the setup of the C-test difficulty prediction task as investigated by Beinborn (2016). C-tests are gap filling exercises proposed by Klein-Braley and Raatz (1982). In their proposed gap scheme, every second word is turned into a gap by removing the latter half of its characters. In contrast to cloze tests, C-tests do not require any distractors, since the first half of the word remains as a hint. Solving C-tests requires orthographic, morphologic, syntactic, and semantic competencies as well as general vocabulary knowledge (Chapelle, 1994). C-tests can be easily created automatically by choosing an arbitrary text and introducing the gaps as described above. Because of the context and the kept word prefixes, C-test gaps typically only allow for a single solution (given by the original text) and therefore do not require manual correction. The biggest challenge, however, lies in controlling the difficulty of the text and the derived C-test with its gaps as we have shown in previous work (Lee et al., 2019).

System objective. Given a large pool $\mathcal{X}$ of C-tests $x \in \mathcal{X}$ with n gaps $g_i \in x$, $1 \leq i \leq n$, the system objective is to learn a classifier $d(g) \in L_D$ to judge the *gap difficulty* of gaps $g \in x$ with minimal training data. As the difficulty classes L_D, we use the four labels *very easy*, *easy*, *hard*, and *very hard* proposed by Beinborn (2016). These four classes are based on the mean error rates $e(g)$ of a gap g observed across all users. Figure 2 shows the mapping between the mean error rates $e(g)$ and the four gap difficulty classes L_D.

Data. For our experiments, we obtained 3,408 solutions to English C-tests from our university's language center. Each participant solved five C-tests with 20 gaps each (i.e., 100 gaps per solution). The five C-tests vary across the participants based on a set of 74 different C-tests in total. We filter out answers from 22 participants who either did not provide any correct answer or only filled out the first of the five C-tests. Based on this dataset, we derive the ground-truth labels for the gap difficulty classification $d(g)$ based on figure 2.

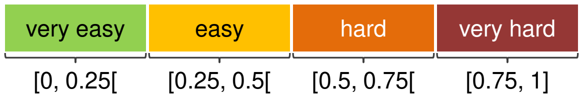


Figure 2: Gap difficulty classes and error rate ranges

Aggregated instances. In contrast to Beinborn's (2016) work, a particular challenge of our setup is the need to *aggregate instances*. The active learning strategies $s(v)$ always sample entire C-tests $x \in \mathcal{X}$ and judge their appropriateness for a user v based on $A(x, v)$. The underlying classifier $d(g)$, however, operates at the level of gaps $g \in x$ within a C-test. Similarly complex setups can be found in multiple other real-world tasks, including educational applications (e.g., providing reading recommendations at book or chapter level, but estimating appropriateness at word or sentence level) and product recommendation tasks (e.g., training a classifier for cast, plot, and action aspects, but recommending entire movies).

For our instantiation, we measure the classifier's uncertainty using the Shannon entropy

$$H(g) = -\sum_{\ell \in L_D} P(\ell \mid g) \log P(\ell \mid g) \qquad (6)$$

across the four difficulty classes L_D of a gap g. $P(\ell \mid g)$ denotes the probability of the classifier d to assign the difficulty class ℓ to gap g. We then aggregate the resulting scores similar to the *total token entropy* proposed by Settles and Craven (2008):

$$U_{\text{ent}}(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{H(g_i)}{H_{\text{max}}} \qquad (7)$$

where H_{max} is the maximum achievable Shannon entropy, which serves as a normalization term. H_{max} can be pre-computed as:

$$H_{\text{max}} = -\sum_{i=1}^{|L_D|} \frac{1}{|L_D|} \log \frac{1}{|L_D|} \qquad (8)$$

User objective. To model the demands of the users, we define five *proficiency levels* L_P =

Level	1	2	3	4	5
Score (%)	0–54	55–64	65–74	75–84	85–100
Users	814	607	724	769	472

Table 1: Proficiency levels, corresponding scores (% correctly filled gaps), and number of users per level.

$\{1, 2, 3, 4, 5\}$ based on the users' ability to solve C-tests. The user representation $\phi(v) \in L_P$ of user v thus returns a proficiency level between 1 and 5 with 5 indicating the highest proficiency.

In our experiments, we use the C-test dataset introduced above to obtain $\phi(v)$. Note that in this dataset, each user solved exactly five C-tests. We therefore map their score (i.e., the percentage of correctly filled gaps) to a proficiency level that roughly corresponds to the language courses offered by the university language center. Table 1 shows the five levels with their corresponding score ranges and the number of users in the dataset.

We estimate the proficiency level of a C-test $x = g_1, g_2, \ldots, g_n$ with

$$f(x) = \psi\left(\frac{1}{n}\sum_{i=1}^{n} c(g_i)\right) \quad (9)$$

where $c\colon g \mapsto \{0, 1\}$ is an indicator function to predict if gap g_i will be correctly (1) or incorrectly (0) answered and ψ maps the percentage of correct answers to the corresponding proficiency level according to Table 1. For our experiments, we define

$$c(g) = \begin{cases} 1 & \text{if } k < j \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

where $k \sim \mathcal{U}(\frac{\ell-1}{|L_P|}, \frac{\ell}{|L_P|})$ and $j \sim \mathcal{U}(0, 1)$ are uniformly sampled random variables and $\ell = d(g)$. Based on our estimation $f(x) \in L_P$, we can now define the error function err as the normalized distance of $f(x)$ to the required proficiency:

$$\mathrm{err}[f(x), \phi(v)] = \frac{1}{|L_P|}|f(x) - \phi(v)| \quad (11)$$

5 Experimental Setup

System setup. We initialize our system with an empty set of labeled instances. In every iteration, we sample a C-test consisting of 20 gaps from the pool of unlabeled instances $\mathcal{X}$ using one of the sampling strategies introduced in the previous section. Then, we obtain labels based on how the user solved the test, which contributes (1) to the overall difficulty prediction for each gap and (2) to the representation of the current user's proficiency.

Our approach can be used with any underlying classifier $d(g)$. In this paper, we train a multilayer perceptron (MLP) to predict the four difficulty classes for a C-test gap. To represent the input of the MLP, we use the 59 features previously proposed by Beinborn (2016). We furthermore introduce two novel features computed from BERT (Devlin et al., 2019): We hypothesize that the masking objective of BERT which masks individual words during training is very similar to a gap filling exercise and thus, a model trained in such a way may provide useful signals for assessing the difficulty of a gap. For each gap, we generate a sentence where only the gap is replaced by the masking token and fetch its predictions from the BERT model. From these predictions we take the prediction probability of the solution as the first feature and the entropy of the prediction probabilities of the top-50 predicted words as the second feature in concordance with findings by Felice and Buttery (2019) who show that entropy strongly correlates with the gap difficulty. Adding both features to the 59 features proposed by Beinborn (2016) increases the accuracy of our MLP from 0.33 to 0.37.[4]

While Beinborn successfully used support vector machines (SVM) in her work, we find that MLPs perform on par with SVMs (for the old and new features) and that they are more robust regarding the choice of the first sampled instance. Moreover, in our initial experiments with little training data, SVMs and Logistic Regression classifiers were only able to predict the majority class.

Our MLP has a single hidden layer consisting of 61 hidden units. We train the neural network for 250 epochs with early stopping after 20 epochs without any improvement and use *Adam* (Kingma and Ba, 2015) as our optimizer. Note that our main interest is in the analysis of the novel active learning approach, which is why we do not systematically study the underlying classifier, but use a setup comparable to the state-of-the-art results reported by Beinborn (2016).

We run experiments for each of our sampling strategy. We select five C-tests without any overlap between users, texts, and their corresponding user answers to create an independent test set and put the remaining 69 C-tests into the pool of unlabeled data. In the first iteration, we use the randomly initialized weights of our neural network to select

[4]The results are averaged across ten runs with different random initializations.

the starting example. To provide comparable results between different runs, we keep the parameter initialization of our neural network fixed when comparing different sampling strategies. We limit each experimental run to $8 \cdot 5 = 40$ iterations, as the five proficiency levels are not evenly distributed with the smallest class having only eight C-tests. At each iteration, we train our model on 80% of the already labeled data and use the remaining 20% as our validation set (split randomly). We use the best-performing model on the validation set for testing and store it as our model initialization for the next iteration. On an *Intel Core i5-4590*, a single run with 40 iterations takes less than four minutes.

Learner behavior. To study the benefit of our approach for different types of learners,[5] we derive four prototypical learner behaviors from our C-test dataset. To prepare this, we first compile a probabilistic model for the learners of each proficiency group as described in Table 1 to obtain learner-specific gap error rates $e(g, v)$. The learner-specific gap error rates are computed by binning all learners into the specific groups and then computing the error rate by averaging for each gap. If there is no error rate for a given gap and learner in our dataset, we use the averaged gap error rate of the corresponding proficiency group to simulate an answer.

Using these learner-specific gap error rates, we predict whether an answer to a C-test gap g is correct or incorrect similar to Equation (10):

$$\hat{c}(g) = \begin{cases} 1 & \text{if } e(g, v) < j \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

In contrast to Equation (10), we do not sample k, but use the learner-specific error rates $e(g, v)$ for gap g_i from the proficiency level $\phi(v)$. Again, $j \sim \mathcal{U}(0, 1)$ is a uniformly sampled random variable.

For a language learning platform, it is likely that motivated learners who continually practice improve their proficiency over time. Less motivated learners or learners who suffer from distractions, interruptions, or frustration, however, may show different paces in their learning speed or even deteriorate in their proficiency. Therefore, we study four prototypical types of learner behavior:

- *Static learners* (STAT) do not improve their skills over the course of our experiments. Instead, they provide answers constantly at the same, pre-defined proficiency level. This models learners with a slow progress or with little motivation overall.
- *Motivated learners* (MOT) continually improve their language proficiency throughout our experiments with a fixed step size of t_1 C-tests. That is, we simulate that their proficiency level $\phi(v)$ increases by one every t_1 iterations.
- *Interrupted learners* (INT) experience a drop in their proficiency during our experiments. Such cases occur, for example, if a learner has to interrupt their learning process for a longer time. For our simulation, we start with the motivated learner setup, constantly increasing the proficiency every t_1 iterations. However, this learner experiences a sudden increase (t_2) and drop (t_3) in the proficiency level by one. After recovering from the drop (t_4) the proficiency will again increase according to the motivated learner (t_5).
- *Artificially decreasing learner.* (DEC) Finally, our last group of simulated learners displays a constant drop in their proficiency during our simulation. Although such cases rarely occur in the real world, we use this learner to evaluate all sampling strategies in the case of constant drop. Similar to the motivated learner, we start with the highest possible proficiency and decrease it by one every t_1 iterations.

For our experiments, we assume a static learner that remains at proficiency level $\phi(v) = 3$. For motivated learners, we set the initial proficiency level to 1 and use a step size of $t_1 = 8$, so that they traverse all proficiency levels throughout a single run. For interrupted learners, we also use $t_1 = 8$ with an additional increase after $t_2 = 12$, a drop after $t_3 = 16$, and a recovery (increase) after $t_4 = 20$. Starting from $t_5 = 24$, interrupted learners behave the same as motivated learners.

Like Beinborn (2016), we cannot publish the C-test data due to data privacy reasons, but we provide our code and simulated learner models on GitHub.[6]

6 Experiments

We present and discuss our results for U_{ent} and A as defined in section 4. For each strategy we run our experiments ten times with different weight

[5] Henceforth, we use *learner* to refer to the users of an educational application rather than to a machine learning system.

[6] `https://github.com/UKPLab/acl2020-empowering-active-learning`

initializations and report the averaged scores. For random sampling, we do ten runs with different random seeds for each weight initialization to provide more stable results. We set $\lambda = 0.5$ for our trade-off sampling strategy.

6.1 Evaluation metrics

As our system and user objectives have different scopes (gap-level vs. exercise-level), we quantify both differently. To measure the system objective, we report the *accuracy* of our model for predicting the individual gap difficulties of the test data after each iteration. As our training data increases by 20 gaps after each iteration, we provide plots for all experiments from the first to the last (40-th) iteration. For quantifying the user objective, we evaluate all sampling strategies across all 40 iterations, i.e., how well our sampling strategies were able to satisfy the user's needs after the whole set of exercises. Instead of accuracy, we take the distance-based metric *mean absolute error* (MAE). As users explicitly query a C-test of a specific proficiency level at each iteration, suggesting a C-test which deviates by two levels from the requested proficiency has a worse impact on the user's learning experience than a C-test which only deviates by one level. For better interpretability, we do not normalize the MAE as we do for our error function err, i.e., a MAE of 1 means that on average, the difficulty of the sampled instances was off by a whole proficiency level from the queried ones.

6.2 Results

Since the interrupted learner experiences both a drop and increase in proficiency in a less constant manner than the motivated or decreasing learners, we conduct further analysis of our sampling strategies for the interrupted learner.

System objective. Figure 3 shows the system objective for U_{ent} after each iteration. Vertical blue lines indicate increases in the learner's proficiency whereas the vertical yellow line indicates a drop. We observe that although random sampling performs rather well in the early iterations, all our proposed strategies as well as the uncertainty sampling baseline are able to outperform it in the later iterations. Moreover, all proposed strategies perform similar to uncertainty sampling. This is surprising, especially for the user-oriented sampling strategy as it inherently does not optimize the system objective. One reason for this may be the similarity

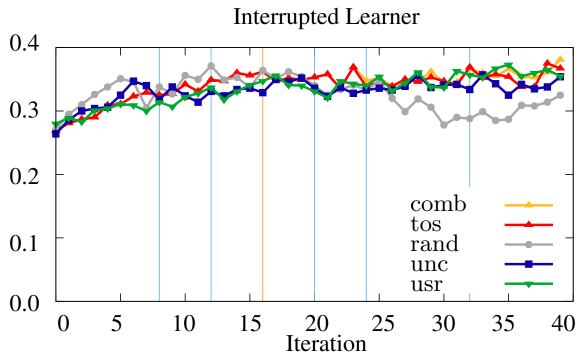


Figure 3: Accuracy on the test data for U_{ent}.

	STAT	MOT	INT	DEC
tos	**.344**	.338	.339	.327
comb	.343	**.340**	**.341**	.327
usr	.338	.331	.334	.328
unc	.332	.331	.331	**.331**
rand	.325	.325	.325	.325

Table 2: Averaged accuracy over all iterations for U_{ent}

of the user-oriented sampling strategy to *curriculum learning* (Bengio et al., 2009), which opts to organize model training in a meaningful way. As we sample instances the model is most confident in (i.e., have the highest prediction confidence) this leads to instances which are easier to learn and may especially be helpful in low-data scenarios.

To better quantify our results, we compare the averaged accuracy scores across all iterations, shown in table 2 and conduct Wilcoxon signed-rank tests (Wilcoxon, 1992) on the active learning curves for system and model objectives to test for statistical significance. We can observe that for the static, motivated, and interrupted learners both our joint sampling strategies outperform all baselines significantly ($p < 0.05$), but show no significant difference between each other.[7] Only for the decreasing learner all strategies show no significant difference at all. In concordance with our observations for the user-oriented sampling which may benefit from first sampling easy-to-learn instances, jointly optimizing system and user objective seems to benefit from curriculum learning and active learning paradigms.

User objective. Table 3 shows the MAE for all strategies using U_{ent}. We can observe that all strate-

[7]The system performance of random sampling remains the same for all learner types as it is averaged across all runs.

	STAT	MOT	INT	DEC
tos	0.98	0.65	0.93	0.75
comb	0.98	0.63	0.88	**0.65**
usr	**0.85**	**0.58**	**0.65**	0.75
unc	1.17	1.33	1.35	1.72
rand	1.16	1.22	1.82	1.24

Table 3: MAE for U_{ent}

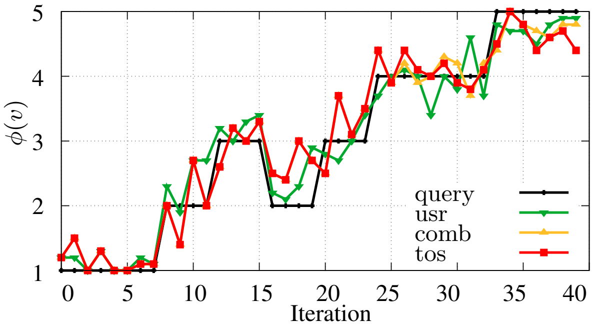


Figure 4: Sampled instances for the interrupted learner.

gies which consider a separate user objective sample instances which significantly better fit the current user proficiency.[8] Furthermore, the combined sampling approach which puts more emphasis on the user objective outperforms our trade-off sampling for all learner behaviors and even manages to outperform the user-oriented sampling strategy for the decreasing learner.

We further investigate how well our approaches react to changes in the user objective by plotting the mean difficulty $\phi(v)$ of sampled instances after each step for all our strategies modeling the user objective. As figure 4 shows, all sampling strategies are able to match the queried C-test difficulties well, as they do not deviate much from the queried difficulty (in black).

Adaptive choice of λ. We furthermore investigate how the choice of λ affects our trade-off sampling strategy. As the system predictions may not be very accurate in early iterations, it is reasonable to put more emphasis on the system objective in the beginning, but focus on providing suited C-tests (user objective) in later iterations. We thus define λ as an adaptive function $\lambda = f(i) = \frac{1}{\sqrt{i}} = i^{-0.5}$ which highly emphasizes the system objective in early stages and anneals with an increasing number of iterations i.

[8] Statistical testing was again conducted using a Wilcoxon signed-rank test for $p < 0.05$.

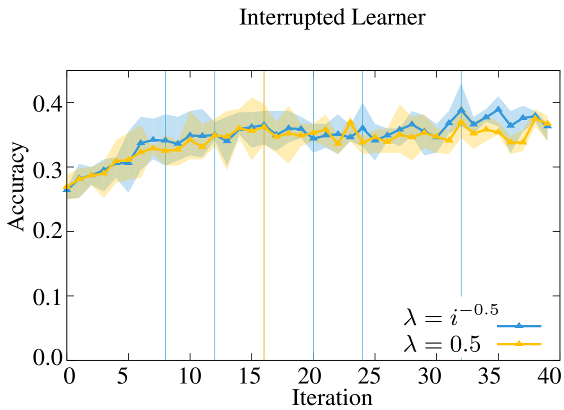


Figure 5: Accuracy of tos for annealed and fixed λ.

Acc	STAT	MOT	INT	DEC
tos_λ	.333	**.346**	**.347**	.314
tos	.334	.338	.339	.327
MAE	STAT	MOT	INT	DEC
tos_λ	**0.85**	**0.53**	**0.48**	**0.53**
tos	0.98	0.65	0.93	0.75

Table 4: Averaged accuracy scores and MAE with an annealed λ for U_{ent}.

Figure 5 shows the system performance of our trade-off sampling strategy averaged across ten different runs. The colored areas show the corresponding upper and lower quartiles. As shown in table 4, we can see that our annealed λ leads to considerable improvements for system and user objective, leading to a significant increase in average accuracy from 0.339 to 0.347 and a decrease in the MAE from 0.93 to 0.48 for the interrupted learner, outperforming all other sampling strategies.

Further findings. We observe similar results for system and user objectives for the other learner types. Investigating the stability of all sampling approaches furthermore shows that our joint optimization strategies perform better and more stable in early iterations.

Due to averaging, U_{ent} cannot distinguish between C-tests with only a few highly uncertain gaps and C-tests which have a higher number of less uncertain gaps. However, in preliminary experiments with a different aggregation function which is more robust to C-tests with only a few highly uncertain gaps, we come to similar findings across all sampling strategies and learner types. Detailed results for our other learner behaviors, the stability of our sampling strategies, and the results of our preliminary experiments with a different aggregation function are provided in the paper's appendix.

Limitations. Although our setup with simulated learners may seem artificial compared to an evaluation study with real-world learners, to conduct such a study in an ethical way, we need to ensure that participants are not hurt in their learning process. Thus, strategies which can be evaluated in user studies are limited to those which consider the user objective. In contrast, the use of simulated learners allows us to compare our proposed strategies against common active learning strategies which do not consider the user objective at all.

Another limitation is how to estimate a learner's current proficiency given that we do not know the true difficulty of a C-test. This raises the general question of using relative or absolute difficulties for the selection of suited exercises. In this work, we assumed absolute proficiency levels and implemented according learner behaviors to provide a more controlled environment for our experiments. In the case of absence of any absolute (true) difficulty estimations for C-tests, we see several directions for future work:

a) As a simple baseline, a normalized version of $\psi(x)$ may be applied on a learner's previously filled-out C-tests. However, this assumes that all C-tests are equally difficult which may lead to unsuited C-tests.

b) Training an additional model for assessing a learner's proficiency given their results on a C-test with the gap-difficulty predictions from our model serving as additional input.

c) Instead of using the absolute difficulty, one may define an optimal error margin as a zone of proximal development (Vygotsky, 1978). This requires an adaptation of the user objective to the relative difficulties of exercises for individual learners, but may be an important step in achieving highly personalized user models without any absolute labels.

7 Conclusion

In this work, we investigated how we can incorporate user feedback into existing active learning approaches without hurting the user's actual needs. We formalize both *system* (active learning) and *user* objectives and propose two novel sampling strategies which aim to maximize both objectives jointly. We evaluate our sampling strategies for the task of selecting suited C-tests, a type of fill-the-gap exercise, which fit the current proficiency of a human learner. We create simulated learners for five different proficiency levels from real-world data and use them to define different learning behaviors. Our experiments show that both our novel sampling strategies are successfully selecting instances which lead to a better model training while not hurting a learner's progress by selecting too easy or too difficult C-tests. Although system and user objective at first seem counteracting, our experiments indicate that they complement each other as jointly optimizing them outperforms optimizing only one of the goals. Additional experiments with an adaptive λ for our trade-off sampling strategy show that properly balancing system and user objective can lead to considerable improvements in performance for both objectives.

Our findings open up new opportunities for training models on low-resource scenarios with implicitly collected user feedback while jointly serving the user's actual needs. Additional use cases like the training of personalized recommendation models as well as the use of reinforcement learning to find a good trade-off between system and user objective remain to be investigated in future work.

Acknowledgments

This work has been supported by the German Research Foundation with the ArguAna project (GU 798/20-1) and the Evidence project (GU 798/27-1). We thank the anonymous reviewers for their detailed and helpful comments as well as Edwin Simpson and Yevgeniy Puzikov for the insightful discussions about our work. We especially thank the language center of the Technische Universität Darmstadt for providing us with the data and Dr. Lisa Beinborn for providing us with the code to extract her proposed features.

References

Lisa Marina Beinborn. 2016. *Predicting and Manipulating the Difficulty of Text-Completion Exercises for Language Learning*. Ph.D. thesis, Technische Universität Darmstadt, Darmstadt.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 41–48, New York, NY, USA. ACM.

Nhabhat Chaimongkol, Shotiga Pasiphol, and Sirichai Kanjanawasee. 2016. Computerized Adaptive Testing with Reflective Feedback: A Conceptual Framework. *Procedia-Social and Behavioral Sciences*, 217:806–812.

C. A. Chapelle. 1994. Are C-tests valid measures for L2 vocabulary research? *Second Language Research*, 10(2):157–187.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL/HLT)*, pages 4171–4186, Minneapolis, MI, USA.

Meng Fang, Jie Yin, and Dacheng Tao. 2014. Active learning for crowdsourcing using knowledge transfer. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI)*, pages 1809–1815, Québec, Canada.

Mariano Felice and Paula Buttery. 2019. Entropy as a proxy for gap complexity in open cloze tests. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP)*, pages 323–327, Varna, Bulgaria.

Yang Gao, Christian M Meyer, and Iryna Gurevych. 2018. APRIL: Interactively Learning to Summarise by Combining Active Preference Learning and Reinforcement Learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4120–4130, Brussels, Belgium.

Peter Hastings, Simon Hughes, and M. Anne Britt. 2018. Active learning for improving machine learning of student explanatory essays. In Carolyn Penstein Rosé, Roberto Martínez-Maldonado, H. Ulrich Hoppe, Rose Luckin, Manolis Mavrikis, Kaska Porayska-Pomsta, Bruce McLaren, and Benedict du Boulay, editors, *Artificial Intelligence in Education (AIED)*, volume 10947 of *Lecture Notes in Computer Science*, pages 140–153. Cham: Springer.

Andrea Horbach and Alexis Palmer. 2016. Investigating Active Learning for Short-Answer Scoring. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications (BEA)*, pages 301–311, San Diego, CA, USA.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, CA, USA.

Christine Klein-Braley and Ulrich Raatz. 1982. Der C-Test: ein neuer Ansatz zur Messung allgemeiner Sprachbeherrschung. *AKS-Rundbrief*, 4:23–37.

Florian Laws, Christian Scheible, and Hinrich Schütze. 2011. Active Learning with Amazon Mechanical Turk. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1546–1556, Edinburgh, UK.

Ji-Ung Lee, Erik Schwan, and Christian M. Meyer. 2019. Manipulating the difficulty of c-tests. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 360–370, Florence, Italy.

David D. Lewis and William A. Gale. 1994. A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 3–12, Dublin, Ireland.

Olana Missura and Thomas Gärtner. 2011. Predicting dynamic difficulty. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2007–2015. Curran Associates.

Avinesh P.V.S and Christian M. Meyer. 2017. Joint optimization of user-desired content in multi-document summaries by learning from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1353–1363, Vancouver, Canada.

Ishan Rastogi, Aditya Kanade, and Shirish Shevade. 2018. Active learning for efficient testing of student programs. In Carolyn Penstein Rosé, Roberto Martínez-Maldonado, H. Ulrich Hoppe, Rose Luckin, Manolis Mavrikis, Kaska Porayska-Pomsta, Bruce McLaren, and Benedict du Boulay, editors, *Artificial Intelligence in Education (AIED)*, volume 10947 of *Lecture Notes in Computer Science*, pages 296–300. Cham: Springer.

Tobias Scheffer, Christian Decomain, and Stefan Wrobel. 2001. Active Hidden Markov Models for Information Extraction. In Frank Hoffmann, David J. Hand, Niall Adams, Douglas Fisher, and Gabriela Guimaraes, editors, *Advances in Intelligent Data Analysis (IDA)*, volume 2189 of *Lecture Notes in Computer Science*, pages 309–318. Berlin/Heidelberg: Springer.

Burr Settles. 2012. *Active Learning*, volume 18 of *Synthesis Lectures on Artificial Intelligence and Machine Learning*. Morgan & Claypool.

Burr Settles and Mark Craven. 2008. An Analysis of Active Learning Strategies for Sequence Labeling Tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1070–1079, Honolulu, HI, USA.

Claude Elwood Shannon. 1948. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. 2008. Cheap and Fast – But is it Good? Evaluating Non-Expert Annotations for Natural Language Tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 254–263, Honolulu, HI, USA.

Huong May Truong. 2016. Integrating learning styles and adaptive e-learning system: Current developments, problems and opportunities. *Computers in human behavior*, 55(Part B):1185–1193.

Lev Vygotsky. 1978. *Mind in society: The development of higher psychological processes*. Cambridge: Harvard University Press.

Shiyu Wang, Haiyan Lin, Hua-Hua Chang, and Jeff Douglas. 2016. Hybrid Computerized Adaptive Testing: From Group Sequential Design to Fully Sequential Design. *Journal of Educational Measurement*, 53(1):45–62.

Frank Wilcoxon. 1992. Individual comparisons by ranking methods. In *Breakthroughs in statistics*, pages 196–202. Springer.

Yi Zheng and Hua-Hua Chang. 2015. On-the-fly assembled multistage adaptive testing. *Applied Psychological Measurement*, 39(2):104–118.

A Appendices

A.1 Results of U_{ent} for other learner types

Figure 6 shows our results for the static, motivated, and artificially decreasing learner. As with the interrupted learner, blue (yellow) vertical lines indicate an increase (drop) in the learner's proficiency. Similar to the results for the interrupted learner, all strategies outperform random sampling in later iterations.

A.2 An outlier-invariant variation of U

Due to averaging, U_{ent} cannot distinguish between C-tests with only a few highly uncertain gaps and C-tests which have a higher number of less uncertain gaps. We investigated another aggregation function U_{soft} in preliminary experiments, which measures the entropy across all gaps and thus, is more robust to C-tests with only a few highly uncertain gaps.

Formulation. For our second formulation of U, we use a different aggregation method. Due to the mean, U_{ent} is unable to distinguish between C-tests where the system is highly uncertain for only a few gaps and C-tests where all gaps are less, but more equally uncertain. We propose to use the softmax function σ for normalizing $H(g_i)$ and then to compute the entropy across all gaps g_i. U_{soft} thus considers the distribution of gap-uncertainties and favours C-tests with equally distributed gap-uncertainties over C-tests with only a few highly uncertain gaps.

$$U_{\text{soft}}(x) = \gamma\,[-\sum_{i=1}^{n} \sigma_i(H(g_i)) \log \sigma_i(H(g_i))] \tag{13}$$

As the squashing of the individual gap entropy values removes the information about their magnitude, we furthermore scale the resulting value by the normalized mean entropy

$$\gamma = \frac{1}{n \log n} \sum_{i=1}^{n} \frac{H(g_i)}{H_{\max}} \tag{14}$$

for all gaps g_i in the C-test.

Results. Figure 11 shows similar tendencies as we already found for U_{ent} in section 6. Again, we can observe that random sampling performs better in early iterations, while the other sampling strategies outperform it in latter iterations. Averaging the accuracy across all iterations (table 5) shows that both our joint sampling strategies tos and comb again perform in average better than the other sampling strategies for the static, motivated, and interrupted learners. However, conducting a Wilcoxon signed-rank test with $p < 0.05$ shows that the active learning curves only significantly differ for the static learner.

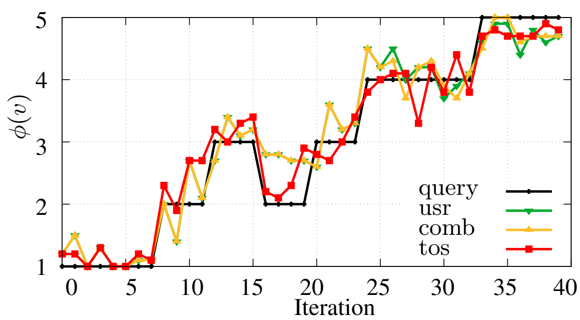


Figure 8: Sampled instances for the interrupted learner using U_{soft}.

For the user objective (also shown in table 5) we observe that all strategies which include a user objective significantly outperform rand and unc, but there is no clear favorite amongst them. This can also be seen in figure 8 where all strategies manage to sample instances close to the queried difficulty (in black).

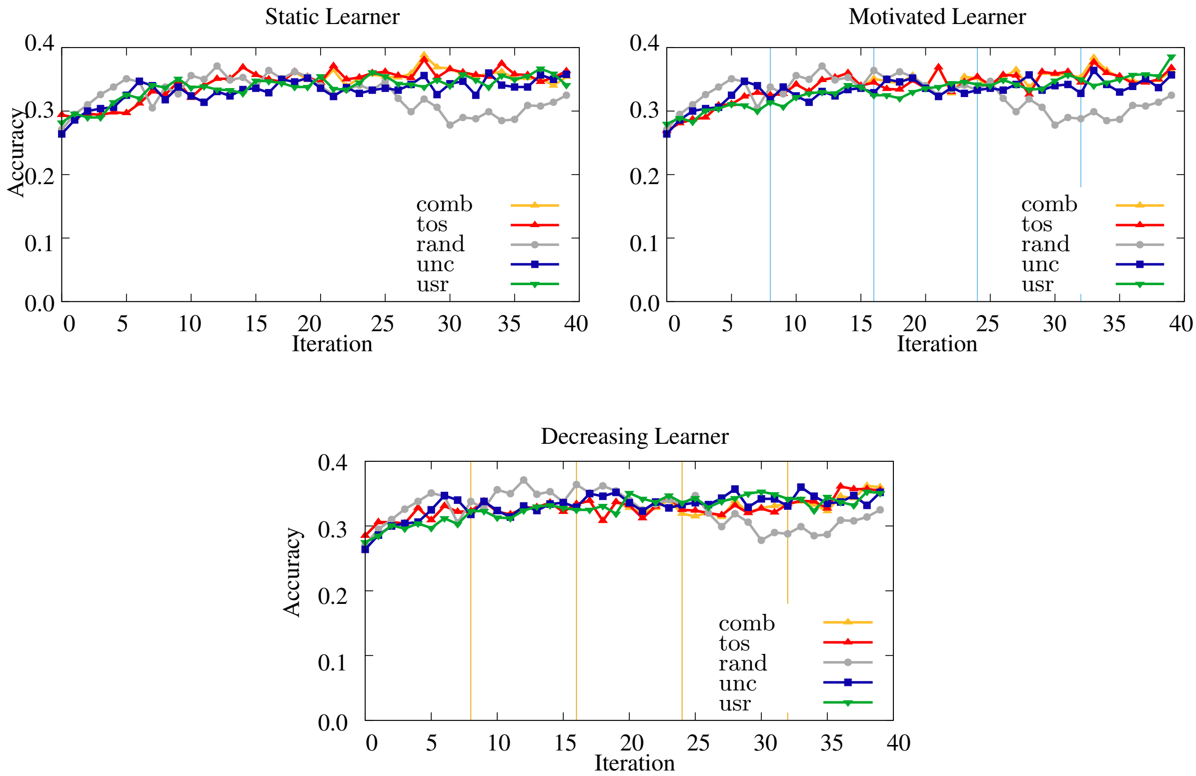


Figure 6: Accuracy scores for the static, motivated, and artificially decreasing learners using U_{ent}.

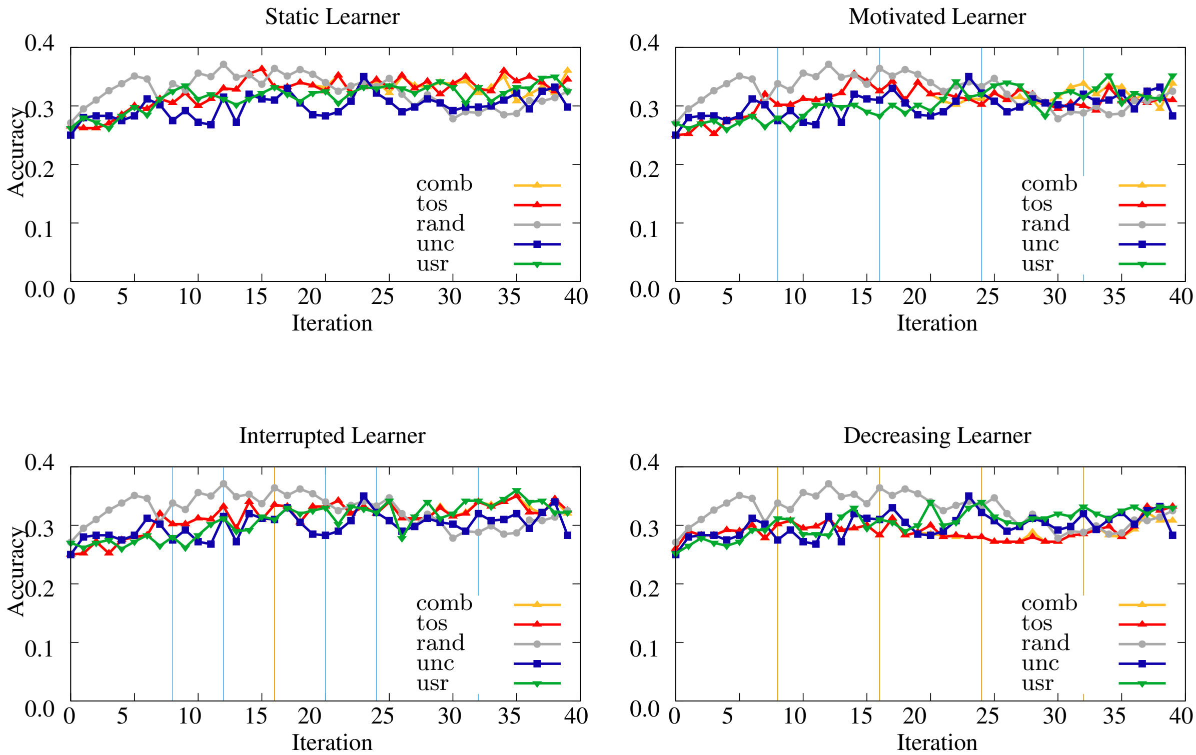


Figure 7: Accuracy on the test data for U_{soft}.

A.3 Impact of the aggregation function

Figure 9 compares both our aggregation functions U_{ent} and U_{soft} against each other on the interrupted learner for uncertainty, combined, and trade-off sampling. Although U_{ent} and U_{soft} differ to some regard, directly comparing both aggregation functions and the respective aggregated scores (cf., table 5 shows that there is no clear favourite between both. Extensive work with respect to both aggregation functions as well as additional aggregation strategies remains to be investigated in future work.

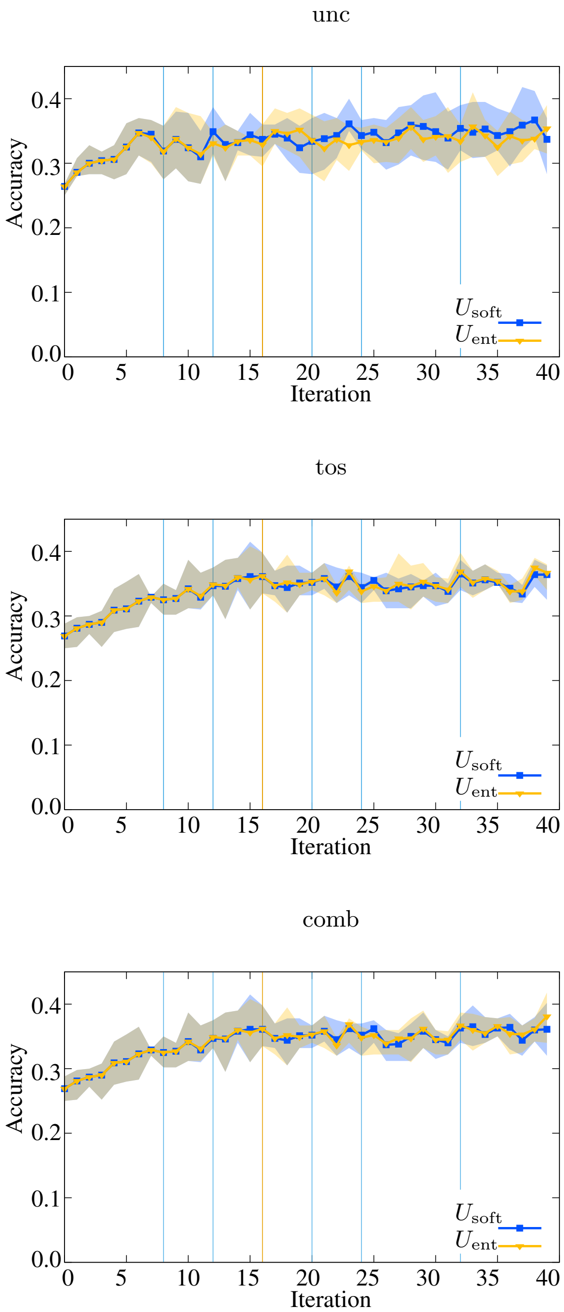


Figure 9: Comparing U_{ent} and U_{soft} for the interrupted learner.

A.4 Stability of system objective

To provide estimates how stable our approaches are across different randomly initialized weights, we compute the upper and lower quartiles for each sampling strategy across all runs. Figures 10 and 11 show our results for the interrupted learner.

Overall, we observe that user-oriented sampling has lower deviations across different runs for both our aggregation functions U_{ent} and U_{soft}. One reason for this may be that in contrast to uncertainty sampling, we query instances with highly certain predictions in our user-oriented sampling approach. This leads to sampled instances which are easier to learn resulting in a higher training stability with small data. Comparing the user-oriented against our joint sampling strategies shows that especially in the earlier iterations, our proposed sampling strategies perform better and provide more stable training.

A.5 Further investigation of λ

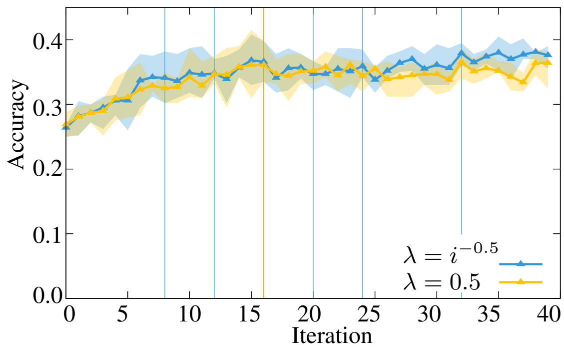


Figure 12: Accuracy of trade-off sampling for annealed and fixed λ using U_{soft} for the interrupted learner.

To further validate our findings for an annealed λ, we conduct the same experiments with our novel aggregation function U_{soft}. As with U_{ent}, we obtain significant improvements for our trade-off sampling strategy (figure 12) for the motivated and interrupted learner, but also a significant decrease for the static and decreasing learner. With respect to the user objective, we do not see any significant differences at all, indicating that U_{soft} does not benefit at all from the emphasised user objective in later iterations.

Table 5 (including the previous results for better comparability) shows the results for all learner behaviours and both our aggregation functions U_{ent}

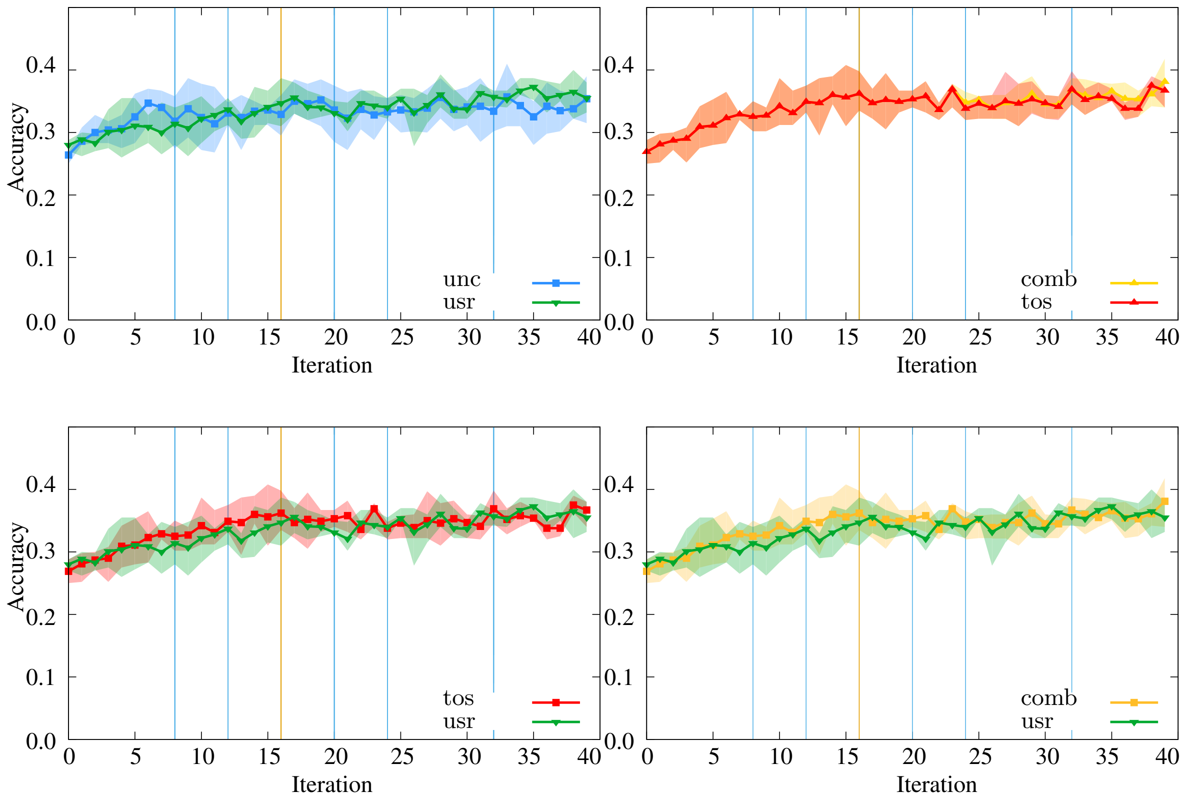


Figure 10: Upper and lower quartiles for the interrupted learner using U_{ent}.

	$\mathrm{U_{ent}}$								$\mathrm{U_{soft}}$							
	Accuracy				MAE				Accuracy				MAE			
	STAT	MOT	INT	DEC	STAT	MOT	INT	DEC	STAT	MOT	INT	DEC	STAT	MOT	INT	DEC
tos_λ	.333	.346	.347	.314	0.85	0.53	0.48	0.53	.331	.345	.347	.316	0.86	0.64	0.59	0.70
tos	.334	.338	.339	.327	0.98	0.65	0.93	0.75	.345	.336	.338	.327	0.91	0.64	0.62	0.70
comb	.343	.340	.341	.327	0.98	0.63	0.88	0.65	.344	.338	.340	.326	0.93	0.63	0.62	0.66
usr	.338	.331	.334	.328	0.85	0.58	0.65	0.75	.337	.331	.334	.328	0.92	0.63	0.59	0.70
unc	.332	.331	.331	.331	1.17	1.33	1.35	1.72	.336	.336	.336	.335	1.24	1.32	1.31	1.72
rand	.325	.325	.325	.325	1.16	1.22	1.82	1.24	.325	.325	.325	.325	1.16	1.22	1.82	1.24

Table 5: Averaged accuracy and MAE for all strategies (including the annealed λ strategy) for U_{ent} and U_{soft}.

and U_{soft}. As can be seen, using an annealed λ (tos_λ) leads to the best results with respect to the user objective for U_{ent} but fails to improve the results for U_{soft}.

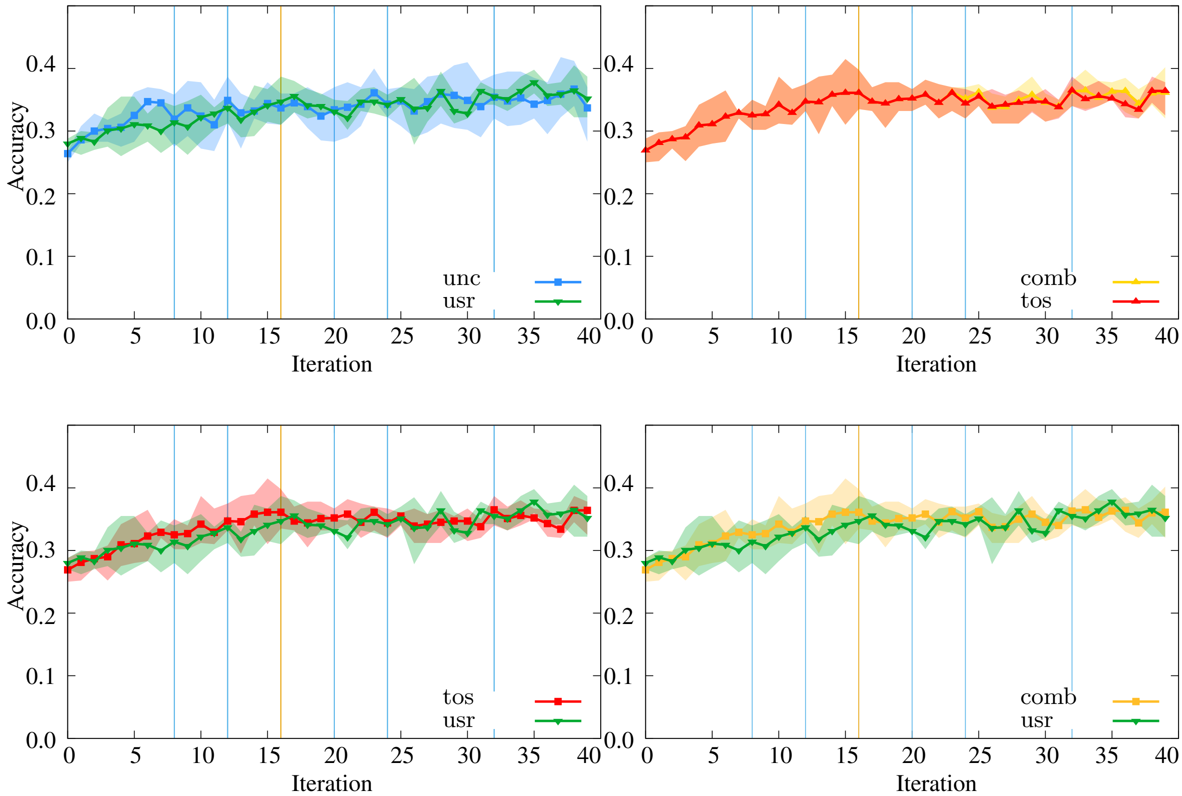


Figure 11: Upper and lower quartiles for the interrupted learner using U_{soft}.

Chapter 10

TexPrax: A Messaging Application for Ethical, Real-time Data Collection and Annotation

TexPrax: A Messaging Application for Ethical, Real-time Data Collection and Annotation

Lorenz Stangier[*] [†] and **Ji-Ung Lee**[*] [†] and **Yuxi Wang**[‡] and **Marvin Müller**[‡]
Nicholas Frick[‡] and **Joachim Metternich**[‡] and **Iryna Gurevych**[†]
Ubiquitous Knowledge Processing (UKP) Lab[†]
Institute of Production Management, Technology and Machine Tools (PTW)[‡]
Department of Computer Science and Hessian Center for AI (hessian.AI)[†]
Department of Engineering[‡]
Technical University of Darmstadt

Abstract

Collecting and annotating task-oriented dialog data is difficult, especially for highly specific domains that require expert knowledge. At the same time, informal communication channels such as instant messengers are increasingly being used at work. This has led to a lot of work-relevant information that is disseminated through those channels and needs to be post-processed manually by the employees. To alleviate this problem, we present TexPrax, a messaging system to collect and annotate *problems*, *causes*, and *solutions* that occur in work-related chats. TexPrax uses a chatbot to directly engage the employees to provide lightweight annotations on their conversation and ease their documentation work. To comply with data privacy and security regulations, we use an end-to-end message encryption and give our users full control over their data which has various advantages over conventional annotation tools. We evaluate TexPrax in a user-study with German factory employees who ask their colleagues for solutions on problems that arise during their daily work. Overall, we collect 202 task-oriented German dialogues containing 1,027 sentences with sentence-level expert annotations. Our data analysis also reveals that real-world conversations frequently contain instances with code-switching, varying abbreviations for the same entity, and dialects which NLP systems should be able to handle.[1]

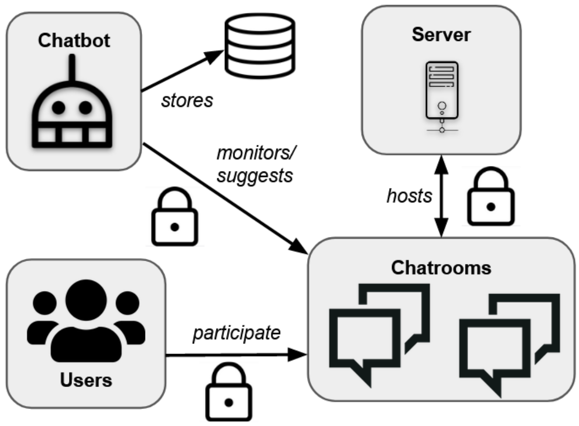


Figure 1: Overview of TexPrax. All users as well as the chatbot communicate via chatrooms that are hosted on a Synapse server instance. All messages are end-to-end encrypted using the Matrix communication protocol.

1 Introduction

The lack of annotated data—especially in languages other than English—is one of the key open challenges in task-oriented dialogue processing (Razumovskaia et al., 2022). This becomes even more challenging for very task-specific application domains with only a small number of experts that are sufficiently qualified to generate dialogue data or provide annotations (Sambasivan et al., 2021). At the same time, using informal communication channels such as instant messengers at work has become increasingly popular (Rajendran et al., 2019; Newman and Ford, 2021). Although this can accelerate troubleshooting, most of the knowledge that is communicated informally may be lost without an additional error tracking process; which in turn increases documenting work for employees (Müller et al., 2021a). Whereas this could be alleviated by NLP-based assistance systems—that for instance automatically identify *problems*, their *cause*, and their *solution*—they cannot be built without any annotated data. Our goal is to provide an application (TexPrax) to bridge the gap between the lack of annotated task-oriented dialogue data and the increasing need for NLP-based documenting assistance.

Figure 1 provides a high-level overview of TexPrax and all involved parties. Our system communicates as a chatbot that acts as the user interface and recording service at the same time. For the server that hosts the messaging application, the bot

[*]Equal contribution
[1]Code and data are published under an open source license: https://github.com/UKPLab/TexPrax

Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing: System Demonstrations, pages 9–16
November 20–23, 2022. ©2022 Association for Computational Linguistics

server that hosts the messaging application, the bot appears as an additional user and hence, inherits all privileges and restrictions a user can have; including (1) reading any messages written in a chatroom, (2) being invited and removed by the chatroom moderator, and (3) being able to send messages in chatrooms it was invited to. We use privilege (3) to provide label suggestions from a pre-trained model and collect annotations via a reaction mechanism (Figure 5b); attaining a lightweight annotation process with minimal overhead. We also integrate TexPrax via the REST web API into an internal dashboard to automatically store recognized errors as a first step of the error documentation process.

Directly involving employees in data annotation and curation introduces four key advantages over previous approaches that involve crowdsourcing (Crowston, 2012) or use expert annotation tools such as INCEpTION (Klie et al., 2018). First, they are the very domain experts that hold qualified conversations which concern exactly the target-domain. This allows us to directly collect the dialog data instead of having to generate it semi-automatically or asking crowdworkers who can only provide limited expertise (Raghu et al., 2021). Second, the employees have an immediate benefit from annotating and improving the recommendation model as a dashboard integration saves time they would otherwise have to spend on documenting errors later on (hence, an intrinsic motivation). Third, they have full control over their own data which saves time for NLP practitioners as it alleviates research data management. Finally, the use of an end-to-end encryption protocol ensures that only parties selected by the employees will have access to the data even if the server is breached.[2] Our contributions are:

1. An application for collecting and annotating dialogues in real-time to assist employees during their work. To comply with data privacy and safety regulations such as the GDPR (EU, 2016), TexPrax further has received full clearance by the ethics committee and staff council of TU Darmstadt.

2. A German dataset with 202 dialogues, consisting of 591 turns, and 1,027 annotated sentences collected from a highly specific domain, namely an assembly line in a factory.

[2]Upon creating a chat room, they will explicitly be asked if our chatbot is allowed to join the chatroom (opt-in).

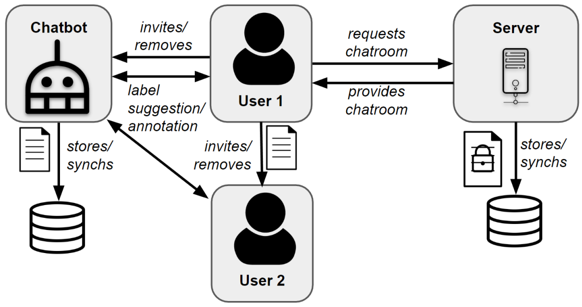


Figure 2: Information flow and privileges between users, server, and chatbot. While staying in a chatroom, the chatbot can decrypt all messages and stores them locally. Messages passed via the server are always encrypted.

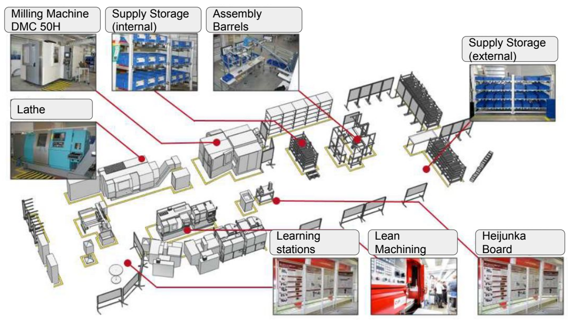


Figure 3: Workstations and machines in the Center for Industrial Productivity (CiP).

2 Use Case

In this work, we focus on assisting employees on the shop floor (the production area in a factory). Our goal is to improve shop floor management (Hertle et al., 2017); a systematic approach for solving processing problems. To efficiently solve such problems, shop floor management defines performance indicators which are used to detect deviations and identify problems which are also used to quantify their successful rectification.

2.1 Production Environment

Our working environment is the learning factory *Center for Industrial Productivity* (CiP) at the Technical University of Darmstadt (TUDa). The factory consists of various assembly stations, machines, and demonstrators (cf. Figure 3) and is run and maintained by ∼15 research assistants and 40 student assistants (Müller et al., 2021b). One substantial challenge is the on-site support in case of problems that occur during their daily work, for instance, if a machine suddenly stops working. Usually, workers then ask their co-workers, technical support support, or the supervising research assistant (who may not be present) for assistance, often via informal communication channels. While this leads to a quick fix of the issue, the knowledge of how to resolve such errors is not explicitly stored and hence, can be forgotten or lost over time.

2.2 Preliminary Survey

To assess the need of an NLP-based assistance system, we rely upon the analysis from a previous survey that was conducted at the CiP (Müller et al., 2021a). In this survey, they identify eight key issues and challenges from an employee's perspective. (1) The most frequently used communication channel are emails. (2) Most questions are answered fast, but in case of a slow return-rate it takes very long to receive an answer which leads to a substantial delay of the assembly line. (3) There are no platforms that pool already encountered problems and solutions. Thus, there is a high demand for such a system. (4) Most employees would use such an application only for work communication. (5) A majority of employees are convinced that such an application could help in substantially reducing the required time to find a solution. (6) Most employees are fine with using such an application on their private phone. (7) All employees agreed to have a chatbot in a group chat monitoring the chatroom, but most stated that this would influence their communication behavior. (8) The most important benefit would be the improvement of knowledge management.

For companies, they identify three important criteria. (1) A high level of data security is essential to avoid any leakage of information outside the company (i.e., the application should be self-hostable). (2) No personal data may be processed to avoid legal complications. (3) The most important benefit would be the improvement of error-reporting and -monitoring processes.

3 System Description

As shown in Figure 1, TexPrax involves three key parties: the users (employees), the chatbot and the server (e.g., hosted by a company). Users communicate via chatrooms; each chatroom including at least two (for a private conversation) or more (for a group conversation) users. Every message a user sends into a room can be read by any other user in the same room. The server is responsible for handling new incoming messages and the distribution of outgoing messages, as well as keeping track of currently active conversations and users. Finally, the chatbot is responsible for monitoring conversations, suggesting labels, and storing the relevant data (locally or in an external database).

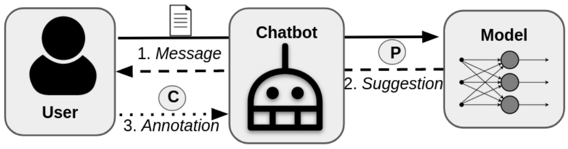


Figure 4: Information flow between the user, chatbot, and the underlying model.

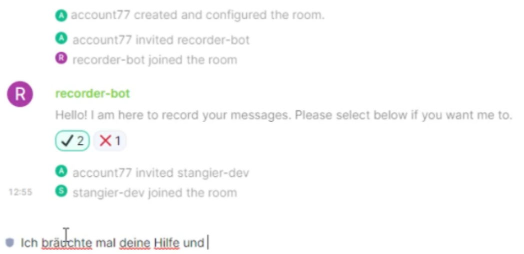
3.1 Interaction and Privileges

A key focus of TexPrax lies within giving users full control over their data and when their conversation should be monitored. We thus provide them with the option to remove the chatbot from a conversation at any time. Moreover, the matrix communication protocol allows users to modify and remove their messages which are then propagated to other participants in a chatroom including the chatbot. This provides a safer communication space to users as they have full control over what messages are stored. To comply with GDPR regulations (EU, 2016), we further implement a feature to obtain the informed consent of the users for each chatroom.

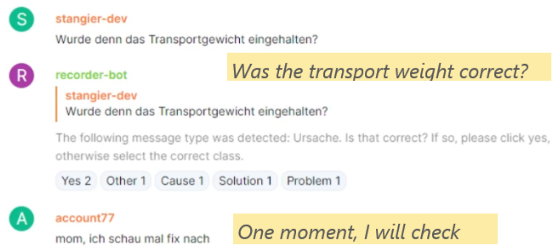
Upon being invited to a chatroom, the chatbot automatically sends an introductory message and explicitly asks if this room shall be recorded (Figure 5a). The user can then respond to the question with one of the provided reactions. If no reaction is selected but a message is sent, the chatbot will assume that the invitation was not intended and leaves the chatroom automatically without recording any message. Only upon acceptance, the chatbot will notify the user and start monitoring and reacting to new messages (Figure 5b). Note, that the chatbot can be removed by the user at any time and invited back in later. Due to the end-to-end encryption, the chatbot will not be able to read any messages that have been sent while it was not present.

Annotations are then made by the user by accepting the suggested label, or providing a correction.[3] Only when a reaction is provided, the message and its class are stored in the internal database.

[3] We also investigated labeling messages using free-text replies; however, users asked for an easier way of interaction.



(a) Introductory message after a new room has been created. If at least one user is against recording, the bot will leave the room.



(b) Label suggestion for a recognized cause.

Figure 5: Example messages of TexPrax.

3.2 Server

The server is based on the Synapse implementation of the matrix protocol[4]; an open-source privacy-centric messaging protocol that enables end-to-end encrypted communication while allowing the server to be hosted on custom hardware (Ermoshina et al., 2016). This guarantees that all messages that are passed between users (and the chatbot) remain encrypted on the server and thus, cannot be read even if the server is breached. The usage of Synapse further allows users to use different client applications such as Element[5] across different platforms (i.e., mobile, desktop, and browser) to send and receive messages. For the study and debugging purposes, we further extended the existing implementation to automatically send an invitation to the chatbot every time a new chatroom is created (users will still be asked for their consent before recording any messages). TexPrax is setup on a virtual machine with 4 CPU cores, 8 GB RAM, and 50 GB of storage.

3.3 Chatbot

The chatbot is based on the nio project[6]—a client library for the matrix protocol—written in Python. As soon as the user allows the chatbot to record messages, it will store every new message including the annotation into a local database. Processing messages can freely be extended; for instance, it is also possible to send the messages to an external instance via HTTP instead storing them locally. To provide the system with additional flexibility, the chatbot can be hosted completely separate from the server. It is thus possible to run different chatbots for each chatroom on different hardware, which can be helpful to better comply with data privacy regulations. As shown in Figure 4, we utilize a pre-trained model to provide users with label suggestions. The chatbot will then react to a message with a label suggestion and ask the user to confirm or correct the notification (they can also just ignore the message). All user annotations are stored separately from the model's suggestion.

4 Data Collection

In contrast to our previous work that investigates expert-annotated named entity recognition (Müller et al., 2021b), our goal is to provide a first solution for collecting annotated data and providing assistance with a minimal effort for users. We thus focus on sentence-level annotations that can be easily provided using message reactions and are suitable for existing shop floor management processes.

4.1 Annotation Task

Following existing workflows for shop floor management that are currently done on paper, we identify three crucial classes for our use case:

1. Problem (**P**): The description of a deviation from an expected target state, e.g., machine breakdowns, material delays, incorrect production processes etc. (often formulated as a question).

2. Cause (**C**): The assumed cause of a problem.

3. Solution (**S**): The action to eliminate the root cause of the problem or to help in finding the possible causes and countermeasures.

4. Other (**O**): None of the above classes (e.g., unrelated messages).

To train an initial label suggestion model, we re-annotated the existing dataset with sentence-level

[4] `https://matrix.org/docs/projects/server/synapse`
[5] `https://element.io/`
[6] `https://matrix.org/docs/projects/sdk/matrix-nio`

Part	Dialogues (D)	Turns (T)	T/D	Problem	Solution	Cause	Other	Total	Sents/D
P1	81	246	3.04	127	74	50	302	553	6.83 ± 3.82
P2	97	309	3.19	117	56	114	145	432	4.45 ± 2.11
P3	24	36	1.50	23	12	1	6	42	1.75 ± 0.66
Total	202	591	2.60	267	142	165	453	1,027	5.08 ± 3.28

Table 1: The number of dialogues, turns, and their ratio (left) and the class distribution on a sentence-level (right).

annotations.[7] This was done by three of the authors that are responsible for managing the CiP. Each of them annotated one third of the dataset and cross-examined all other annotations for possible errors or disagreement. Upon disagreeing on a label, all annotators discussed the respective instance to agree upon the best suited one.

4.2 Participants

All participants were student assistants, technical support staff, or researchers that worked in the CiP and are employed at the university; receiving payment according to the official wages (above German minimum wage). They were informed about the purpose of the study in advance, and provided their informed consent before participation. They further received instructions about how to use the application including the features allowing them to modify and remove already sent messages. Participation was strictly voluntary and anonymous; to further obfuscate the identity of our participants, we created a pool of user accounts from which an account was randomly assigned to each user. For data publication we obfuscate the user accounts by hashing the ID of each user. Overall, our study had a total number of 10 participants over the whole duration (October 2021 to July 2022).

4.3 Data Analysis

Table 1 shows the statistics of the collected data. We split the data into three parts; first, the re-annotated dataset that was used to train the label suggestion model (cf. Section 5), second, data collected between October 2021 and June 2022, and third, the data collected in July 2022 to evaluate the our final system which we also use as the test data for our experiments. The second and third batch of data was each collected in a separate chatroom. An overview of the dialogue properties can be found in Table 1. Overall, the dataset consists of 202 dialogues with 591 turns and 1,027 sentences. A close inspection of the data reveals interesting properties (e.g., grammatically incorrect language, abbreviations, etc.). Despite that, we want to emphasize that there was no single case where our participants could not understand a message.

[7] Deploying TexPrax without any suggestion model does not affect the number of reactions provided by users.

Distributional shifts. Table 1 shows varying class distributions across all three splits. One reason for this may be the amount of expertise in chatrooms across different periods of data collection. For instance, between the first and second part of the data collection which were ~10 months apart, there had been a partial change of staff in work force. With new people joining the CiP, we find a higher number of responses looking for potential causes of a problem, but with less success (i.e., less solutions). We further find that the more acquainted workers in the first data collection tend to provide longer explanations and engage themselves more in chitchat which is reflected in the substantially higher number of *Other* class sentences and a higher sentence-per-dialog ratio (Sents/D).

Slang. We find various occurrences of text mimicking spoken language involving grammatically incorrect expressions. For instance, our participants frequently used *ne* instead of *eine* (*Eng.:* a/an) or as the short form of *nein* (*Eng.:* no).

Abbreviations. We find that our participants tend to communicate in short messages that involve abbreviations. While some are easily understandable for native German speakers—e.g., *vllt.* for *vielleicht* (*Eng.:* maybe)—others such as *V8* for *Variante 8* (a product type) or *wimi* for *wissenschaftliche Mitarbeitende* (*Eng.:* researcher) are highly dependent on the domain.

Filler words. Similar to in-person conversations, we also find an abundance of filler words such as *ah*, *hmm*, and *oh*.

Code switching. We find that participants sometimes tend to code switch from German to English (Scotton and Ury, 1977); especially for short, one word responses (e.g., *Nice!*, *Sorry!*).

5 Experiments

We conduct experiments to gain first insights on how well recent models can perform for providing label suggestions for our use case in future studies.

5.1 Experimental Setup

We evaluate two models that are capable of processing German texts as our baselines. First, the XLMR-base model (Conneau et al., 2020) provided by Huggingface (Wolf et al., 2020) that has been shown to have a solid performance across various languages (Malmasi et al., 2022). Second, a German version of BERT (GBERT, Chan et al. 2020). This has been shown to work well for German tweets that have a similar format (i.e., short, German sentences containing informal language) as our messages (Beck et al., 2021). For sentence classification, we use the [CLS] token to predict if a given sentence states a *problem* (P), a *cause* (C), a *solution* (S), or *other* (O). Across all experiments, we train our models for 10 epochs with a learning rate of $2e^{-5}$ and weight decay of 0.01, and a batch size of 16. We use the parts 1 and 2 as presented in Table 1 for training and use part 3 as the most recently collected dataset for testing.

	P1		P2		P1 + P2	
Model	Acc	F1	Acc	F1	Acc	F1
XLMR	0.357	0.216	0.524	0.315	0.476	0.269
GBERT	0.405	0.267	0.310	0.237	0.429	**0.361**

Table 2: Accuracy and macro-averaged F1 scores of both models trained on different temporal datasplits.

5.2 Results

Table 2 shows the results of both models on the P3 data (cf. Table 1). Both models are not able to achieve a marco-averaged F1 score higher than 0.4, showing that even recent language-specific models struggle for sentence classification when applied to a very specific domain and little training data (432–553 sentences). Interestingly, GBERT outperforms XLMR when trained on P1 data as well and when trained on P1 + P2 data in terms of F1 score. Although we initially conjectured that XLMR may be capable of better handling the code switched data, this does not seem to be the case. We further find that the suggested labels from the GBERT model (trained on P1 data) during the collection of P2 achieved an accuracy of 0.683. While this is a moderately high performance, this also implies that 31.7% of the labels needed to be corrected by our participants.

5.3 Usability

To ensure that this did not substantially impact the usability of TexPrax, we asked our voluntary participants to answer the system usability scale (SUS) questionnaire (Brooke, 1996) upon finishing the final round of data collections (P3). SUS quantifies the relative usability with respect to existing benchmarks and ranges from $\mathbf{A}^+$ (84.1–100 SUS) to **F** (0–51.6 SUS) (Lewis and Sauro, 2018). Overall, seven users participated in the usability study. On average, TexPrax receives a system usability scale score of 81.76 with a standard deviation of 5.46, which indicates an **A** level (80.8–84.0 SUS) usability. We thus conclude that TexPrax achieves a high usability despite the label corrections.

6 Conclusion

We presented TexPrax, a system for collecting annotations and assisting employees by directly engaging them as domain-experts during their daily work. TexPrax allows users to exchange, modify, and delete end-to-end encrypted messages at any time, and an opt-in chatbot to ensure a high level of data privacy and security. We evaluate TexPrax in an assembly line at a learning factory (CiP) where we find that daily work communication is noisy, but efficient and very problem-oriented. While existing models still have difficulties to provide the correct label suggestion, TexPrax still maintains a high usability. We conjecture that TexPrax could be especially beneficial to collect data and build assistance systems in domains with a high share of remote work, such as in software development. For future work, we plan to extend TexPrax to identify and suggest solutions for recognized problems and adapt it to new domains, such as our institute's reading group chat where researchers discuss papers relevant for their research.

Acknowledgements

We thank all subjects from the user study and our anonymous reviewers, Max Glockner, and Jan-Christoph Klie for their helpful feedback. This work has been funded by the European Regional Development Fund (ERDF) and the Hessian State Chancellery – Hessian Minister of Digital Strategy and Development under the promotional reference 20005482 (TexPrax).

Ethics Statement

The collection of data from group- and private-chats requires careful consideration about what kind of data is to be expected and how users can control it. To ensure an ethical data collection and usage, we worked closely together with the respective bodies of our university (TUDa) for developing our final workflow. We want to emphasize that such data should never be collected without the explicit and informed consent of the users. Our participants voluntarily participated in this study and furthermore, had an active interest in the system as they could directly benefit from it.

Pre-study clearance from respective bodies. After defining our data collection workflow and annotation task, we hence asked the ethics committee of our university for ethical clearance.[8] To further ensure the (mental) safety of our participants who were employees of TU Darmstadt, we further asked our university's staff council for their clearance.[9] Both bodies provided their full clearance to conduct this study after minor modifications of the initial workflow involving the account distribution to participants (cf. Section 4.2). Both clearance letters for the final study setup can be shared upon request (in German).

Informed consent. All our participants were fully informed about the data collection processes, for what purpose the data was collected, and how it will be used and released (including the surveys). They all provided their informed consent before requesting an anonymous user account for participation in the study (this was a mandatory requirement from the ethics committee and staff council).

Limitations

Interactive assistance. In this work, we focused on data collection and annotation from workers in a factory environment. Although the integration of TexPrax into their existing dashboard[10] alleviates their daily work, additional assistance could be provided by automatically suggesting solutions for identified problems.

Other use cases. While TexPrax received clearance by our university's ethics committee and staff council, it must be noted that this does not automatically transfer to new use cases or even similar ones at different universities/factories. It is crucial to get at least clearance of the respective staff council before deploying TexPrax to avoid any legal issues that may otherwise arise. Moreover, for the collected data to be of use for the NLP community, the company (or a respective organization) must be willing to share their data publicly. This however implies that deploying TexPrax in organizations that handle sensitive data (e.g., security-related or personal user data) can alleviate the work of employees, but will not result in datasets that can be publicly shared.

Different annotation tasks. The current version of TexPrax is designed as a tool for collecting data and annotations on a sentence-level. Explicitly asking for free-text responses could be one solution to tackle different kinds of annotations such as identifying named entities—for instance, a user could reply to a message containing a named entity by repeating it—however, this may hurt usability and lead to a less frequent usage of the application. To extend TexPrax to different annotation tasks one thus first needs to find a good way to interact with the user.

Propagating dataset changes in trained models. Finally, a last limitation is updating the training data that is implicitly stored in the trained model. The lack of efficient methods to update only specific information in trained models can lead to a substantial overhead when implementing changes in the data made by a user as the whole model needs to be retrained.

[8] https://www.intern.tu-darmstadt.de/gremien/ethikkommisson/index.en.jsp

[9] https://www.personalrat.tu-darmstadt.de/personalrat_1/index.de.jsp

[10] https://www.sfmsystems.de/

References

Tilman Beck, Ji-Ung Lee, Christina Viehmann, Marcus Maurer, Oliver Quiring, and Iryna Gurevych. 2021. Investigating label suggestions for opinion mining in German covid-19 social media. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–13, Online. Association for Computational Linguistics.

John Brooke. 1996. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7.

Branden Chan, Stefan Schweter, and Timo Möller. 2020. German's next language model. In *Proceedings of the 28th International Conference on Com-*

putational Linguistics*, pages 6788–6796, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Kevin Crowston. 2012. Amazon mechanical turk: A research tool for organizations and information systems scholars. In *Shaping the future of ict research. methods and approaches*, pages 210–221. Springer.

Ksenia Ermoshina, Francesca Musiani, and Harry Halpin. 2016. End-to-end encrypted messaging protocols: An overview. In *International Conference on Internet Science*, pages 244–254. Springer.

EU. 2016. Consolidated text: Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance).

Christian Hertle, Michael Tisch, Joachim Metternich, and Eberhard Abele. 2017. Das darmstädter shopfloor management-modell. *Zeitschrift für wirtschaftlichen Fabrikbetrieb*, 112(3):118–121.

Jan-Christoph Klie, Michael Bugert, Beto Boullosa, Richard Eckart de Castilho, and Iryna Gurevych. 2018. The inception platform: Machine-assisted and knowledge-oriented interactive annotation. In *Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations*, pages 5–9.

James R. Lewis and Jeff Sauro. 2018. Item benchmarks for the system usability scale. *Journal of Usability Studies*, 13(3):158–167.

Shervin Malmasi, Anjie Fang, Besnik Fetahu, Sudipta Kar, and Oleg Rokhlenko. 2022. SemEval-2022 task 11: Multilingual complex named entity recognition (MultiCoNER). In *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*, pages 1412–1437, Seattle, United States. Association for Computational Linguistics.

Marvin Müller, Nicholas Frick, and Joachim Metternich. 2021a. Wissen aus betrieblichen chats nachhaltig nutzen. *wt Werkstattstechnik online*, 111(1-2):93–96.

Marvin Müller, Ji-Ung Lee, Nicholas Frick, Lorenz Stangier, Iryna Gurevych, and Joachim Metternich. 2021b. Extracting problem related entities from production chats to enhance the data base for assistance functions on the shop floor. *Procedia CIRP*, 103:231–236.

Sean A. Newman and Robert C. Ford. 2021. Five steps to leading your team in the virtual covid-19 workplace. *Organizational Dynamics*, 50(1):1–11. Virtual Teams.

Dinesh Raghu, Shantanu Agarwal, Sachindra Joshi, and Mausam. 2021. End-to-end learning of flowchart grounded task-oriented dialogs. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4348–4366, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jason Ariel Rajendran, Hanif Baharin, and Fazillah Mohmad Kamal. 2019. Understanding instant messaging in the workplace. In *International Visual Informatics Conference*, pages 640–652. Springer.

Evgeniia Razumovskaia, Goran Glavaš, Olga Majewska, Edoardo Ponti, and Ivan Vulić. 2022. Natural language processing for multilingual task-oriented dialogue. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 44–50, Dublin, Ireland. Association for Computational Linguistics.

Nithya Sambasivan, Shivani Kapania, Hannah Highfill, Diana Akrong, Praveen Paritosh, and Lora M Aroyo. 2021. “Everyone wants to do the model work, not the data work”: Data Cascades in High-Stakes AI. In *proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–15.

Carol Myers Scotton and William Ury. 1977. Bilingual strategies: The social functions of code-switching. *Linguistics. An International Review La Haye*, (193):5–20.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Chapter 11

Annotation Curricula to Implicitly Train Non-Expert Annotators

Annotation Curricula to Implicitly Train Non-Expert Annotators

Ji-Ung Lee*
UKP Lab / TU Darmstadt

Jan-Christoph Klie*
UKP Lab / TU Darmstadt

Iryna Gurevych
UKP Lab / TU Darmstadt

Annotation studies often require annotators to familiarize themselves with the task, its annotation scheme, and the data domain. This can be overwhelming in the beginning, mentally taxing, and induce errors into the resulting annotations; especially in citizen science or crowdsourcing scenarios where domain expertise is not required. To alleviate these issues, this work proposes annotation curricula, a novel approach to implicitly train annotators. The goal is to gradually introduce annotators into the task by ordering instances to be annotated according to a learning curriculum. To do so, this work formalizes annotation curricula for sentence- and paragraph-level annotation tasks, defines an ordering strategy, and identifies well-performing heuristics and interactively trained models on three existing English datasets. Finally, we provide a proof of concept for annotation curricula in a carefully designed user study with 40 voluntary participants who are asked to identify the most fitting misconception for English tweets about the Covid-19 pandemic. The results indicate that using a simple heuristic to order instances can already significantly reduce the total annotation time while preserving a high annotation quality. Annotation curricula thus can be a promising research direction to improve data collection. To facilitate future research—for instance, to adapt annotation curricula to specific tasks and expert annotation scenarios—all code and data from the user study consisting of 2,400 annotations is made available.[1]

1. Introduction

Supervised learning and, consequently, annotated corpora are crucial for many downstream tasks to train and develop well-performing models. Despite improvements of models trained in a semi- or unsupervised fashion (Peters et al. 2018; Devlin et al. 2019), they still substantially benefit from labeled data (Peters, Ruder, and Smith 2019; Gururangan et al. 2020). However, labels are costly to obtain and require domain experts or a large crowd of non-expert annotators (Snow et al. 2008).

Past research has mainly investigated two approaches to reduce annotation cost and effort (often approximated by annotation time); namely, **active learning** and **label suggestions**. Active learning assumes that resources for annotating data are limited and

∗ Equal contribution

Submission received: 7 June 2021; revised version received: 9 December 2021; accepted for publication: 21 December 2021

1 https://github.com/UKPLab/annotation-curriculum.

aims to reduce the number of labeled instances by only annotating those that contribute most to model training (Lewis and Gale 1994; Settles 2012). This often results in sampled instances that are more difficult to annotate, putting an increased cognitive load on annotators, and potentially leading to a lower agreement or an increased annotation time (Settles, Craven, and Friedland 2008). Label suggestions directly target annotators by providing them with suggestions from a pre-trained model. Although they are capable of effectively reducing the annotation time (Schulz et al. 2019; Klie, Eckart de Castilho, and Gurevych 2020; Beck et al. 2021), they bear the risk of biasing annotators toward the (possibly erroneous) suggested label (Fort and Sagot 2010). Both these shortcomings render existing approaches better suited for domain-expert annotators who are less burdened by difficult annotation instances and are less prone to receiving erroneous label suggestions than non-expert annotators. Overall, we can identify a lack of approaches that (1) are less distracting or biased than label suggestions and (2) can also ease the annotation process for non-expert annotators. Especially, the increasing popularity of large-scale, crowdsourced datasets (Bowman et al. 2015; Sakaguchi et al. 2021) further amplifies the need for training methods that can also be applied in non-expert annotator scenarios (Geva, Goldberg, and Berant 2019; Nie et al. 2020; Rogers 2021).

One key element that has so far not been investigated in annotation studies is the use of a curriculum to *implicitly* teach the task to annotators during annotation. The **learning curriculum** is a fundamental concept in educational research that proposes to order exercises to match a learner's proficiency (Vygotsky 1978; Krashen 1982) and has even motivated training strategies for machine learning models (Bengio et al. 2009). Moreover, Kelly (2009) showed that such learning curricula can also be used to teach learners implicitly. Similarly, the goal of **annotation curricula** (AC) is to provide an ordering of instances during annotation that is optimized for learning the task. We conjecture that a good annotation curriculum can implicitly teach the task to annotators—for instance, by showing easier annotation instances before more difficult ones—consequently reducing the cognitive strain and improving annotation speed and quality. In contrast to active learning, which may result in only sampling instances that are difficult to annotate, they explicitly emphasize the needs of a human annotator and gradually familiarize them with the annotation task. Compared to label suggestions, they are less distracting as they do not bear the risk of providing erroneous suggestions from imperfect models, making them well-suited for non-expert annotation scenarios. Furthermore, AC do not require study conductors to adapt existing annotator training processes or annotation guidelines and hence, can complement their annotation project. To provide a first assessment for the viability of such annotation curricula, we investigate the following three research questions:

RQ1. Does the order in which instances are annotated impact the annotations in terms of annotation time and quality?

RQ2. Do traditional heuristics and recent methods for assessing the reading difficulty already suffice to generate curricula that improve annotation time or quality?

RQ3. Can the generation of annotation curricula be further alleviated by interactively trained models?

We first identify and formalize two essential parts to deploy AC: (1) a "strategy" that defines how instances should be ordered (e.g., by annotation difficulty) and (2) an "estimator" that ranks them accordingly. We instantiate AC with an "easy-instances-first" strategy and evaluate heuristic and interactively trained estimators on three

English datasets that provide annotation time which we use as an approximation of the annotation difficulty for evaluation. Finally, we apply our strategy and its best estimators in a carefully designed user study with 40 participants for annotating English tweets about the Covid-19 pandemic. The study results show that the ordering in which instances are annotated can have a statistically significant impact on the outcome. We furthermore find that annotators who receive the same instances in an optimized order require significantly less annotation time while retaining a high annotation quality. Our contributions are:

C1. A novel approach for training non-expert annotators that is easy to implement and is complementary to existing annotator training approaches.

C2. A formalization of AC for sentence- and paragraph-labeling tasks with a strategy that orders instances from easy to difficult, and an evaluation for three heuristics and three interactively trained estimators.

C3. A first evaluation of AC in a carefully designed user study that controls for external influences including:

a) An implementation of our evaluated annotation curriculum strategies and 2,400 annotations collected during our human evaluation study.

b) A production-ready implementation of interactive AC in the annotation framework INCEpTION (Klie et al. 2018) that can be readily deployed.

Our evaluation of different heuristics and interactively trained models further reveals additional factors—such as the data domain and the annotation task—that can influence their aptitude for AC. We thus appeal to study conductors to publish the annotation order and annotation times along with their data to allow future studies to better investigate and develop task- and domain-specific AC.

2. Related Work

Most existing approaches that help with data collection focus on either active learning or label suggestions. Other researchers also investigate tackling annotation task within the context of gamification and introduce different levels of difficulty.

Active Learning. Active learning has widely been researched in terms of model-oriented approaches (Lewis and Gale 1994; Roy and McCallum 2001; Gal, Islam, and Ghahramani 2017; Siddhant and Lipton 2018; Kirsch, van Amersfoort, and Gal 2019), data-oriented approaches (Nguyen and Smeulders 2004; Zhu et al. 2008; Huang, Jin, and Zhou 2010; Wang et al. 2017) or combinations of both (Ash et al. 2020; Yuan, Lin, and Boyd-Graber 2020). Although several works investigate annotator proficiency—which is especially important for crowdsourcing—their main concern is to identify noisy labels or erroneous annotators (Laws, Scheible, and Schütze 2011; Fang et al. 2012; Zhang and Chaudhuri 2015) or distribute tasks between workers of different proficiency (Fang, Yin, and Tao 2014; Yang et al. 2019). Despite the large amount of research in active learning, only a few studies have considered annotation time as an additional cost variable in active learning (Settles, Craven, and Friedland 2008) and even found that active learning can negatively impact annotation time (Martínez Alonso et al. 2015). Other practical difficulties for deploying active learning in real annotation studies stem from additional hyper-parameters that are introduced, but seldom investigated (Lowell, Lipton, and Wallace 2019). In contrast, AC also work well with simple heuristics, allowing researchers to pre-compute the order of annotated instances.

Label Suggestions. Label suggestions have been considered for various annotation tasks in NLP, such as in part-of-speech tagging for low-resource languages (Yimam et al. 2014), interactive entity-linking (Klie, Eckart de Castilho, and Gurevych 2020) or identifying evidence in diagnostic reasoning (Schulz et al. 2019). Especially for tasks that require domain-specific knowledge such as in the medical domain, label suggestions can substantially reduce the burden on the annotator (Lingren et al. 2014). However, they also inherently pose the risk of amplifying annotation biases due to the anchoring effect (Turner and Schley 2016). Whereas domain experts may be able to reliably identify wrong suggestions and provide appropriate corrections (Fort and Sagot 2010), this cannot be assumed for non-experts. This renders label suggestions a less viable solution to ease annotations in non-expert studies where incorrect label suggestions may even distract annotators from the task. In contrast, changing the ordering in which instances are annotated by using AC is not distracting at all.

Annotation Difficulty.. Although difficulty estimation is crucial in human language learning, for instance, in essay scoring (Mayfield and Black 2020) or text completion exercises (Beinborn, Zesch, and Gurevych 2014; Loukina et al. 2016; Lee, Schwan, and Meyer 2019), it is difficult to achieve in annotation scenarios due to the lack of ground truth, commonly resulting in a post-annotation analysis for model training (Beigman Klebanov and Beigman 2014; Paun et al. 2018). To consider the difficulty of annotated instances, a concept that has recently been explored for (annotation) games with a purpose, is **progression**. It allows annotators to progress through the annotation study similar to a game—by acquiring specific skills that are required to progress to the next level (Sweetser and Wyeth 2005). Although several works have shown the efficiency of progression in games with a purpose (Madge et al. 2019; Kicikoglu et al. 2020) and even in crowdsourcing (Tauchmann, Daxenberger, and Mieskes 2020), this does not necessarily benefit individual workers, as less-skilled workers are either filtered out or asked to "train" on additional instances. Moreover, implementing progression poses a substantial burden on researchers due to the inclusion of game-like elements (e.g., skills and levels), or at minimum, the separation of the data according to difficulty and, furthermore, a repeated evaluation and reassignment of workers. In contrast, reordering instances of a single set according to a given curriculum can already be achieved with low effort and can even be implemented complementary to progression.

3. Annotation Curriculum

We first specify the type of annotation tasks investigated in this work, and then formalize AC with the essential components that are required for generating appropriate annotation curricula. Finally, we instantiate an easy-instances-first strategy and define the estimators that we use to generate a respective curriculum.

3.1 Annotation Task

In this work, we focus on sentence- and paragraph-level annotation tasks that do not require any deep domain-expertise and hence are often conducted with non-expert annotators.[2] Such annotation tasks often use a simple annotation scheme limited to a small set of labels, and have been used to create datasets across various research

2 We discuss AC strategies that may be better suited for domain experts in Section 6.

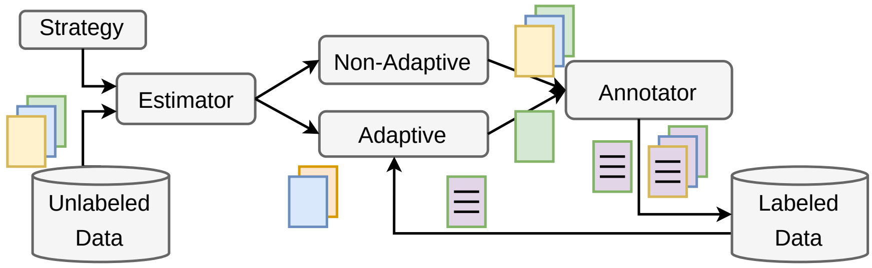


Figure 1: Annotation curricula. First, we define a strategy for ordering instances by annotation difficulty (i.e., easy-first). We then implement estimators that perform the ordering. Estimators can either be non-adaptive (e.g., heuristics) or adaptive (trained models). Finally, annotators receive instances according to the resulting curriculum.

areas, for instance, in sentiment analysis (Pak and Paroubek 2010), natural language inference (Bowman et al. 2015), and argument mining (Stab et al. 2018).

Task Formalization. We define an annotation task as being composed of a set of unlabeled instances $x \in \mathcal{U}$ that are to be annotated with their respective labels $y \in \mathcal{Y}$. We focus on instances x that are either a sentence or a paragraph and fully annotated by an annotator a. Note that for sequence labeling tasks such as named entity recognition, y is not a single label but a vector composed of the respective token-level labels. However, in such tasks, annotations are still often collected for a complete sentence or paragraph at once to provide annotators with the necessary context (Tomanek and Hahn 2009).

3.2 Approach

Figure 1 provides a general overview of AC. Given a set of unlabeled instances $x \in \mathcal{U}$, we define a strategy $\mathcal{S}$ that determines the ordering in which annotated instances should be presented (easy-instances-first). We then specify "adaptive" and "non-adaptive" estimators $f(\cdot)$ that approximate the true annotation difficulty. In this work, we focus on task-agnostic estimators that can easily be applied across a wide range of tasks and leave the investigation on task-specific estimators—which may have higher performance but also require more implementation effort from study conductors—for future work.[3] Depending on the estimator, we then order the annotated instances either beforehand (non-adaptive), or select them iteratively at each step based on the predictions of an interactively trained model (adaptive).

Formalization. Ideally, an annotation curriculum that optimally introduces annotators to the task would minimize (1) annotation effort and (2) error rate (i.e., maximize annotation quality). As the annotation error can only be obtained post-study, we can only use annotation effort, approximated by annotation time, for our formalization; however, we conjecture that minimizing annotation time may also have a positive impact on annotation quality (given that the annotators remain motivated throughout their work). To further reduce noise factors during evaluation, we focus on annota-

3 We discuss some ideas for task-specific estimators in Section 6.

tion studies that involve a limited number of instances (in contrast to active learning scenarios that assume an abundance of unlabeled data). We thus formalize annotation curriculum as the task of finding the optimal curriculum $\mathcal{C}^*$ out of all possible curricula $\mathcal{C}$ (i.e., permutations of $\mathcal{U}$) for a finite set of unlabeled instances $\mathcal{U}$ that minimizes the total annotation time $\mathcal{T}$; namely, the sum of individual annotation times $t_i \in \mathbb{R}^+$ for all instances $x_i \in \mathcal{U}$ with i denoting the i-th annotated instance:

$$\mathcal{C}^* = \arg\min_{\mathcal{C}} \sum_{i=1}^{|\mathcal{U}|} a_i(x_i|x_0 \dots x_{i-1}) \tag{1}$$

where $a_i : \mathcal{U} \rightarrow \mathcal{T}$ describes the annotator after annotating $i-1$ instances.

Strategy. Due to the large number of $n!$ possible curricula $\mathcal{C}$ resulting from $n = |\mathcal{U}|$ instances, solving Equation 1 is intractable for large n even if $a(\cdot)$ was known. We can furthermore only assess the true effectiveness of a curriculum $\mathcal{C}$ post-study, making it impossible to find the optimal curriculum $\mathcal{C}^*$ beforehand. We hence require a strategy $\mathcal{S} \sim \mathcal{C}^*$ that specifies how instances of $\mathcal{U}$ should be ordered optimally. Similar to educational approaches, we rely on estimating the "difficulty" of an instance to generate our curriculum (Taylor 1953; Beinborn, Zesch, and Gurevych 2014; Lee, Schwan, and Meyer 2019). In this work, we investigate an easy-instances-first strategy that has been shown to be a reasonable strategy in previous work (Tauchmann, Daxenberger, and Mieskes 2020); thereby sorting instances in ascending order according to their difficulty. Our $\mathcal{C}^*$ is thus approximated by the ordered set $\mathcal{S} = \{x_1, \dots, x_n | \forall x_{1\leq i \leq n} \in \mathcal{S} : f(x_i) \leq f(x_{i+1})\}$ with $f(\cdot)$ being the difficulty estimator.

Non-adaptive Estimators. We define non-adaptive estimators as heuristics or pre-trained models that are not updated interactively. The respective annotation curriculum can thus be pre-computed and does not impose any additional changes to the underlying annotation platform. To estimate the annotation difficulty, non-adaptive estimators define a scoring function $f_{\bar{a}} : \mathcal{U} \rightarrow \mathbb{R}$. In this work, we evaluate non-adaptive estimators that are commonly used in readability assessment to score the reading difficulty of a text (Xia, Kochmar, and Briscoe 2016; Deutsch, Jasbi, and Shieber 2020). Although they are not capable of capturing any task-specific difficulties, they have the advantage of being applicable to a wide range of tasks with low effort for study conductors. The following heuristics and pre-trained models are investigated to obtain difficulty estimations for the easy-instances-first curriculum:

Sentence Length (sen**)** The number of tokens in a sentence averaged across the whole document (i.e., the number of tokens for single sentence instances).

Flesch-Kincaid (FK**)** A readability score based on the number of words, syllables, and sentences (Kincaid et al. 1975).

Masked Language Modeling Loss (mlm**)** As shown in recent work, the losses of a masked language model may be used to obtain an assessment of text complexity (Felice and Buttery 2019). We use the implementation of Salazar et al. (2020).

Adaptive Estimators. While simple heuristics or annotator-unaware models allow us to pre-compute annotation curricula, they do not consider any user-specific aspect that may influence the difficulty estimation (Lee, Meyer, and Gurevych 2020). Consequently, the resulting curriculum may not provide the optimal ordering for a specific annotator.

To select the instance with the most appropriate difficulty for an annotator $a_i(\cdot)$ at the i-th iteration, we use a model $\theta_i(\cdot) \sim a_i(\cdot)$ that is updated with an increasing number of annotated instances. We conjecture that using $\theta(\cdot)$ to predict the relative difficulty—in contrast to non-adaptive estimators that provide an absolute difficulty estimation—may be more robust to task-specific influences as they are inherited in all instances annotated by $a(\cdot)$. When training adaptive estimators, we use annotation time to approximate the difficulty of a specific instance due to its availability in any annotation scenario. At iteration i, we thus train the model $\theta_i : \mathcal{L} \rightarrow \mathcal{T} \subseteq \mathbb{R}^+$ to predict the annotation times $t \in \mathcal{T}$ for all labeled instances $\hat{x} \in \mathcal{L}$. Similar to active learning, we now encounter a decreasing number of unlabeled instances and an increasing number of labeled instances. The resulting model is then used to estimate the annotation time for all unlabeled instances $x \in \mathcal{U}$. The resulting scoring function is now defined as $f_a : \theta_i, \mathcal{U} \rightarrow \mathbb{R}^+$. Finally, we select instance $x^* \in \mathcal{U}$ with the minimal rank according to f_a.

$$x^* = \arg\min_{f_a} \theta_i(x) \tag{2}$$

Following our strategy $\mathcal{S}$, this results in selecting instances for annotation that have the lowest predicted annotation time. We specifically focus on regression models that can be trained efficiently in-between annotation and work robustly in low-data scenarios. We choose Ridge Regression, Gaussian Process Regression, and GBM Regression.

4. Evaluation with Existing Datasets

To identify well-performing non-adaptive and adaptive estimators, we first evaluate AC on existing datasets in an offline setting. We focus on datasets that provide annotation time which is used to approximate the annotation difficulty during evaluation (to address the lack of gold labels in actual annotation scenarios). Following Settles, Craven, and Friedland (2008), we conjecture that instances with a higher difficulty require more time to annotate. For comparison, we then compute the correlations between different orderings generated according to our easy-instances-first strategy using text difficulty heuristics (non-adaptive) and interactively trained models (adaptive) with the annotation time (approximated annotation difficulty). We evaluate our estimators in two setups:

Full We evaluate how well adaptive and non-adaptive estimators trained on the whole training set correlate with the annotation time of the respective test set (upper bound).

Adaptive We evaluate the performance of adaptive estimators in an interactive learning scenario with simulated annotators and an increasing number of training instances.

4.1 Datasets

Overall, we identify three NLP datasets that provide accurate annotation time for individual instances along with their labels:

Muc7$_\mathbf{T}$ Tomanek and Hahn (2009) extended the MUC7 corpus that consists of annotated named entities in English Newswire articles. They reannotated the data with two annotators A and B while measuring their annotation time per sentence.

SigIE is a collection of email signatures that was tagged by Settles, Craven, and Friedland (2008) with twelve named entity types typical for email signatures such as phone number, name, and job title.

SPEC The same authors (Settles, Craven, and Friedland 2008) further annotated sentences from 100 English PubMed abstracts according to their used language (speculative or definite) with three annotators.

Name	Task	$\|\mathcal{D}\|$	$\|\mathcal{D}_{\mathbf{train}}\|$	$\|\mathcal{D}_{\mathbf{dev}}\|$	$\|\mathcal{D}_{\mathbf{test}}\|$	$\mu_{\|\mathcal{D}\|}$	$\sigma_{\|\mathcal{D}\|}$	μ_t	σ_t
Muc7_T A	ST	3,113	2,179	467	467	133.7	70.8	5.4	3.9
Muc7_T B	ST	3,113	2,179	467	467	133.7	70.8	5.2	4.2
SigIE	ST	251	200	-	51	226.4	114.8	27.0	14.7
SPEC	Cl	850	680	-	170	160.4	64.2	22.7	12.4

Table 1: Annotation task (ST for sequence tagging, Cl for classification) and the number of instances per dataset and split. $\mu_{|\mathcal{D}|}$ denotes the average instance length in characters and μ_t the average annotation time. $\sigma_{|\mathcal{D}|}$ and σ_t denotes the standard deviation, respectively. Across all datasets, annotation time is reported for annotating the whole instance (i.e., not for individual entities).

Table 1 provides an overview of the used datasets. It can be seen that Muc7_T is the largest corpus ($|\mathcal{D}|$); however, it is also the one that consists of the shortest instances on average ($\mu_{|\mathcal{D}|}$). Furthermore, Muc7_T also has the lowest annotation times (μ_t) and a low standard deviation (σ_t). Comparing the number of entities per instance between Muc7_T (news articles) and SigIE (email signatures) shows their differences with respect to their domains with an average number of 1.3 entities ($\sigma = 1.4$) in Muc7_T and 5.3 entities ($\sigma = 3.0$) in SigIE. Moreover, we find that the SigIE corpus has a higher ratio of entity tokens (40.5%) than Muc7_T (8.4%), which may explain the long annotation time. Interestingly, the binary sentence classification task SPEC ("speculative" or "definite") also displays a substantially longer annotation time compared to Muc7_T (on average, more than four times) which may also indicate a higher task difficulty or less proficiency of the involved annotators.

Data Splits. For Muc7_T, we focus on the annotations of the first annotator Muc7_T A; using Muc7_T B yields similar results. For SPEC, we use ALL.DAT for our experiments. None of the aforementioned datasets provide default splits. We hence create 80-20 train-test splits of SPEC and SigIE for our experiments. To identify the best hyperparameters of our adaptive estimators, we split the largest corpus (Muc7_T) into 70-15-15 train-dev-test splits. All splits are published along with the code and data.

4.2 Experimental Setup

Our goal is to evaluate how well the ordering generated by an estimator correlates with the annotation time provided in the respective datasets.

Evaluation Metrics. We evaluate all estimators by measuring Spearman's ρ between the true and generated orderings of all instances in the test data. We obtain the generated ordering by sorting instances according to the predicted annotation time. For our adaptive estimators that explicitly learn to predict the annotation time, we further report the

Name	Features	MAE	RMSE	R^2	ρ	t
RR($\alpha = 0.5$)	BOW	1.85	2.96	0.47	0.73	0.42
RR($\alpha = 0.5$)	S-BERT	1.92	2.84	0.51	0.79	**0.04**
RR($\alpha = 1$)	BOW	1.80	2.91	0.49	0.74	0.41
RR($\alpha = 1$) *	S-BERT	1.89	2.82	0.52	0.79	0.04
GP(kernel=Dot + White)	BOW	1.82	2.93	0.48	0.74	257.67
GP(kernel=Dot + White) *	S-BERT	**1.80**	**2.76**	**0.54**	**0.81**	14.35
GP(kernel=RBF(1.0)	BOW	5.33	6.71	-1.73	-0.12	300.38
GP(kernel=RBF(1.0)	S-BERT	5.33	6.71	-1.73	-0.12	32.66
GBM	BOW	2.07	3.26	0.36	0.68	0.25
GBM *	S-BERT	1.83	2.83	0.52	0.79	2.98

Table 2: Hyperparameter tuning for adaptive estimators. We train on Muc7_T A and evaluate on its development set. t denotes the total time for training and prediction on the whole dataset. Best parameters are marked by * and the best scores are highlighted in **bold**. We report the mean absolute error (MAE), the rooted mean squared error (RMSE), Spearman's ρ, and the coefficient of determination (R^2).

mean absolute error (MAE), the rooted mean squared error (RMSE), and the coefficient of determination (R^2).

Models and Features. For an effective deployment in interactive annotation scenarios, we require models that are capable of fast training and inference. We additionally consider the amount of computational resources that a model requires as they pose further limitations for the underlying annotation platform. Consequently, fine-tuning large language models such as BERT is infeasible as they require long training times and a large amount of computational resources.[4] Instead, we utilize a combination of neural embeddings obtained from a large pre-trained language model combined with an efficient statistical model. As our goal is to predict the total time an annotator requires to annotate an instance (i.e., a sentence or a paragraph), we further require a means to aggregate token- or subtoken-level embeddings that are used in recent language models (Sennrich, Haddow, and Birch 2016). One such solution is S-BERT (Reimers and Gurevych 2019) which has shown a high performance across various tasks. Moreover, Reimers and Gurevych (2019) provide S-BERT for a variety of BERT-based models, allowing future study conductors to easily extend our setup to other languages and specific tasks. For computational efficiency, we use the *paraphrase-distilroberta-base-v1* model, which utilizes a smaller, distilled RoBERTa model (Sanh et al. 2019). As a comparison to S-BERT, we further evaluate bag-of-words (BOW) features for all three models (cf. Table 2). For the Ridge Regression (RR), Gaussian Process Regression (GP), and GBM Regression (GBM) models, we use the implementations of Pedregosa et al. (2011) and Ke et al. (2017).

Hyperparameter Tuning. We use the full experimental setup to identify the best performing parameters for our experiments using simulated annotators. We evaluate different

4 Note that using such models would require an annotation platform to either deploy an own GPU or buy additional computational resources from external providers.

Name	Model	MAE	RMSE	R^2	ρ	t
Muc7_T A	RR	1.87	2.68	0.56	0.80	0.15
	GP	1.79	2.66	0.57	0.82	7.23
	GBM	1.95	2.97	0.47	0.75	3.40
Muc7_T B	RR	2.19	3.42	0.44	0.79	0.02
	GP	2.08	3.37	0.46	0.81	8.85
	LGBM	2.13	3.50	0.41	0.75	2.90
SigIE	RR	7.96	9.50	0.46	0.73	0.00
	GP	7.62	9.60	0.44	0.70	0.08
	GBM	8.22	10.84	0.29	0.55	0.14
SPEC	RR	9.63	13.86	-0.14	0.50	0.03
	GP	7.63	12.07	0.14	0.51	0.73
	GBM	8.05	12.50	0.07	0.35	1.70

Table 3: Performance of the best performing adaptive estimators on the four datasets (Muc7_T provides annotation times from two different annotators A and B) trained on the respective train and evaluated on their test splits. We report the mean absolute error (MAE), the rooted mean squared error (RMSE), the coefficient of determination (R^2) and Spearman's ρ.

values for regularization strength (α) for RR and we evaluate different kernel functions for GP. To ensure that the required training of our adaptive estimators does not negatively affect the annotations due to increased loading times and can be realistically performed during annotation, we further measure the overall training time (in seconds). We use the development split of Muc7_T A to tune our hyperparameters for all models used across all datasets. Considering the small number of training instances in both datasets, we do not tune SigIE- or SPEC-specific hyperparameters. All experiments were conducted using an *AMD Ryzen 5 3600*. Table 2 shows the results of our hyperparameter tuning experiments. Overall, we find that S-BERT consistently outperforms BOW in terms of Spearman's ρ. As the result of the hyperparameter tuning, we use S-BERT embeddings as input features and evaluate GP with a combined dot- and white-noise kernel and RR with $\alpha = 1$ in our adaptive experiments.

4.3 Experimental Results

We first report our experimental results for the full and adaptive setup. For conducting our experiments with simulated annotators, we use the best performing models from our hyperparameter tuning of the respective models on the Muc7_T dataset and report the results of the best performing models.

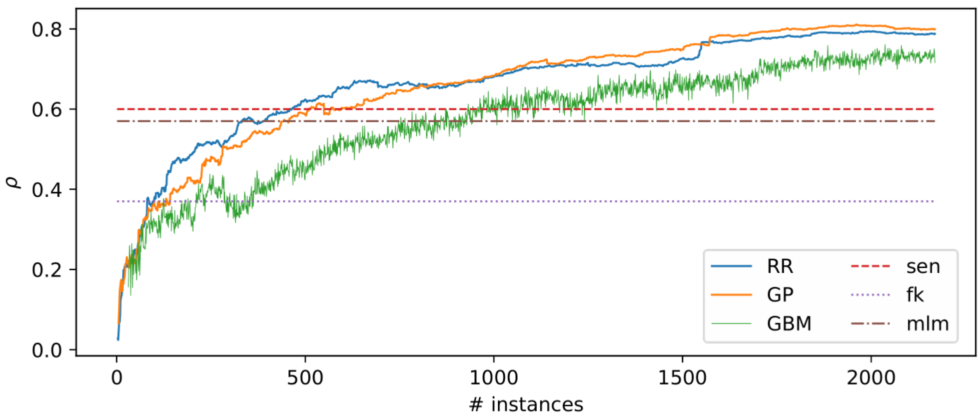
Full Results. Table 4 shows the results for the heuristic estimators and regression models evaluated on the test split of each dataset. We find that heuristics that mainly consider length-based features (sen and FK) are not suited for the SigIE data that consist of email signatures. One reason for this may be the different text type of email signatures in comparison to Newswire articles and PubMed abstracts. More specifically, analyzing the ratio between non-alphabetical or numeric characters (excluding @ and .) and other characters shows that SigIE contains a substantial number of characters that are used for visually enhancing the signature (some are even used in text art). Overall, 29.9% of the characters in SigIE are non-alphabetical or numeric, in contrast to 16.7% in SPEC and 19.9% in Muc7_T.[5] Considering that only 1.7% of them appear within named entities in SigIE (such as + in phone numbers) most of them rather introduce noise especially for length based-features such as sen and FK. On Muc7_T and SPEC, all three heuristics produce an ordering that correlates with annotation time to some extent. On average, mlm is the best performing and most robust heuristic across all three datasets. For our adaptive estimators, RR and GP both similarly outperform GBM in terms of Spearman's ρ. However, we can find that GP consistently outperforms RR and GBM in terms of MAE and RMSE, as well as in terms of R^2 on Muc7_T and SPEC. We report the extensive results in Table 3.

Dataset	sen	FK	mlm	RR	GP	GBM
Muc7_T A	0.60	0.37	0.57	0.80	**0.82**	0.75
Muc7_T B	0.60	0.38	0.55	0.79	**0.81**	0.75
SigIE	0.08	0.01	0.59	**0.73**	0.70	0.55
SPEC	**0.63**	0.38	0.32	0.50	0.51	0.35
Average	0.48	0.29	0.52	0.71	0.71	0.60

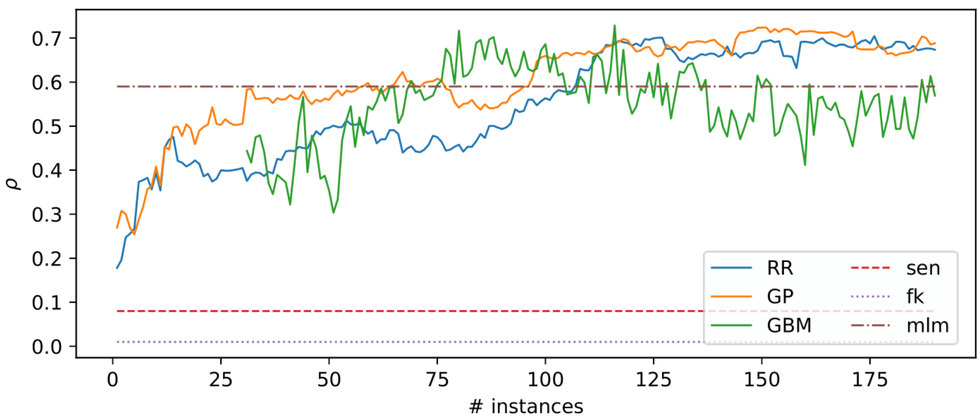
Table 4: Spearman's ρ between test data and the orderings generated by the evaluated heuristics and adaptive models.

Adaptive Results. To evaluate the performance of adaptive estimators with increasing numbers of annotated instances, we perform experiments with simulated annotators. At each iteration, we use a model trained on the already-annotated data to select the instance with the lowest predicted annotation time (randomly in the first iteration). The simulated annotator then provides the respective gold annotation time, which is then added to the training set. Finally, the model is re-trained and evaluated on the test data. These steps are repeated until all instances are annotated. Figure 2 shows the Spearman's ρ performance of all three models after each iteration across all datasets. We can observe that all models display a rather steep learning curve after training on only a few examples, despite suffering from a cold start in early iterations. Moreover, we find that GP and RR are capable of outperforming mlm consistently after 100–500 instances. GBM shows the weakest performance and is consistently outperformed by the other models for Muc7_T and SPEC while being rather noisy. Although we find that non-adaptive estimators can suffice especially in early iterations, our experiments also show the potential of adaptive estimators with an increasing number of annotations. This indicates that hybrid approaches that combine non-adaptive and adaptive estimators could be an interesting direction for future work. For instance, one may consider using non-adaptive estimators in early stages until a sufficient number of annotated instances are available to train more reliable adaptive estimators. Another approach could be to combine the rankings of different estimators, for instance, via Borda count (Szpiro 2010) or learn a weighting of the individual estimators.

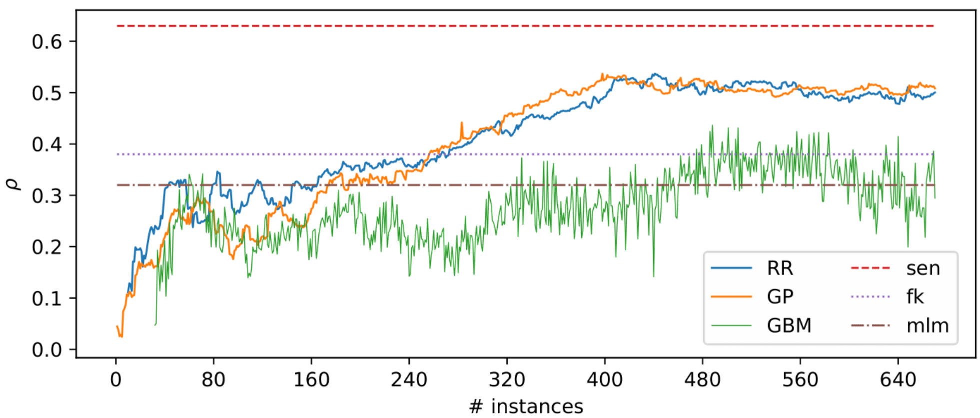
5 The Twitter data we introduce in Section 5 consists of 20.7% non-alphabetical or numeric characters.



(a) Muc7$_T$ A



(b) SigIE



(c) SPEC

Figure 2: Experimental results of our adaptive estimators with simulated annotators. Horizontal lines show the performance of the respective non-adaptive estimators.

5. Human Evaluation

To evaluate the effectiveness of our easy-instances-first AC with real annotators, we conduct a user study on a classification task for English tweets and analyze the resulting annotations in terms of *annotation time* and *annotation quality*. We design the study to not require domain-expertise and conduct it with citizen science volunteers.[6]

Hypothesis. We investigate the following hypothesis: Annotators who are presented with easy instances first and then with instances that gradually increase in terms of annotation difficulty require less annotation time or have improved annotation quality compared with annotators who receive the same instances in a random order.

5.1 Study Design

A careful task and data selection are essential to evaluate AC, as our goal is to measure differences that solely result from a different ordering of annotated instances. We also require instances with varying difficulty, further restricting our study design in terms of task and data.

Data Source. To avoid compromising the study results due to noisy data, we use an existing corpus that has been carefully curated and provides gold labels for evaluating the annotation quality. To involve non-expert annotators, we further require data that do not target narrow domains or require expert knowledge. As such, tasks such as identifying part-of-speech tags would substantially reduce the number of possible study participants due to the required linguistic knowledge. We identify COVIDLies (Hossain et al. 2020) as a suitable corpus due to the current relevance and the high media-coverage of the Covid-19 pandemic; ensuring a sufficient number of participants who are well-versed with the topic. The corpus consists of English tweets that have been annotated by medical experts with one out of 86 common misconceptions about the Covid-19 pandemic. Each instance consists of a tweet-misconception pair and if the tweet "agrees", "disagrees", or has "no stance" toward the presented misconception.

Annotation Task. Using the COVIDLies corpus as our basis, we define a similar task that is better suited for lay people and that allows us to explicitly control the annotation difficulty. We restrict the task to identifying the most appropriate misconception out of six possible choices. Furthermore, we only include tweets that agree with a misconception (i.e., we do not ask for a stance annotation) to avoid interdependencies between stance and misconception annotations that may introduce additional noise to the results and put an unnecessary burden on the participants.[7] To exclude further sources of noise for our study, we manually check all tweets and remove all duplicates (possibly due to retweets) and hyperlinks to increase readability and avoid distractions. We also remove all tweets that were malformed (i.e., ungrammatical or containing several line breaks) or linked to misconceptions with less than five semantically similar candidates that could serve as distractors.[8] For the final selection, we choose the 60 shortest tweets.

6 We provide a statement regarding the conduct of ethical research after the conclusion.
7 We experimented with including stance annotations (positive, negative, or neutral) during early stages of our study setup but removed them due to a substantially increased overall annotation difficulty.
8 The sets of similar misconceptions were manually created as explained in the next paragraph.

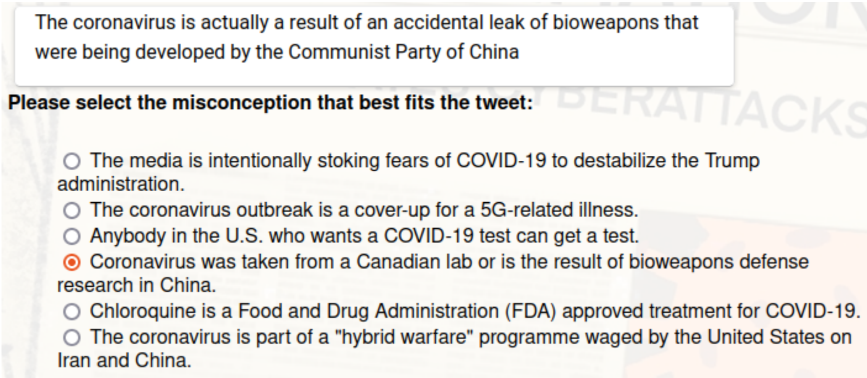
Distractor Selection. The goal of the study is to observe effects that solely result from the ordering of instances with varying annotation difficulty. Hence, we need to ensure that annotated instances correspond to specific difficulties and are balanced equally for each participant. To control the annotation difficulty, we construct five possible sets of misconceptions for each instance that are presented to the annotator; each corresponding to a respective difficulty-level ranging from "very easy" to "very difficult". Each set consists of the expert-selected misconception and five additional misconceptions that serve as distractors which are commonly used in cloze-tests (Taylor 1953). Following existing research on automated cloze-test generation, we focus on **semantic similarity** to generate distractor subsets (Agarwal and Mannem 2011; Mostow and Jang 2012; Yeung, Lee, and Tsou 2019) and manually create one set of five semantically dissimilar and one set of five semantically similar misconceptions for each misconception.[9] As semantically dissimilar distractors are much easier to identify than semantically similar ones (Mostow and Jang 2012), we can manipulate annotation difficulty by adapting the number of semantically similar distractors; that is, starting from the set of dissimilar (very easy) misconceptions, we can gradually increase the difficulty by replacing a dissimilar misconception with a similar one until only the set of similar (very difficult) misconceptions remains. Figure 3 shows a tweet from our user study with its respective easy and difficult misconception sets. As can be seen, the difficult misconception set consists of two more semantically similar misconceptions. Especially notable is the third misconception, which states the opposite of the tweet's misconception but with a similar wording.

5.2 Study Setup

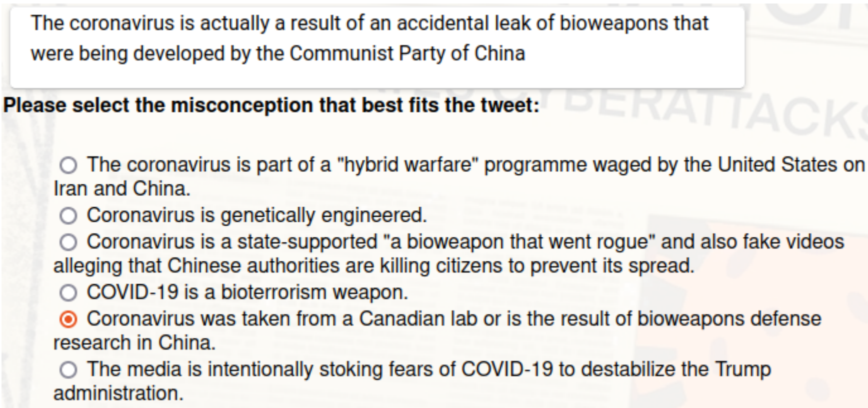
We set up our evaluation study as a self-hosted Web application that is only accessible during the study (one week). Participants can anonymously participate with a self-chosen, unique study key that allows them to request the deletion of their provided data at a later point. Upon registration, they are informed about the data presented and collected in the study, its further use, and the purpose of the study. Before collecting any data, participants are explicitly asked for their informed consent. Overall, we recruited 40 volunteers who provided their informed consent to participate in our study and annotated 60 instances each.

Participants. Our volunteers come from a variety of university majors, native languages, English proficiency, and annotation experience backgrounds. All participants provided a rather high self-assessment of English proficiency, with the lowest proficiency being intermediate (B1) provided by only one participant. Seventy percent of the participants stated an English proficiency-level of advanced (C1) or proficient (C2). Most participants have a higher level of education and are university graduates with either a Bachelor's or Master's degree; however, none of them have a medical background, which may have given them an advantage during the annotation study. Upon completing the annotations, all participants received a questionnaire including general questions about their previous annotation experience and perceived difficulty of the task (cf. Section 5.5).

9 Initially, we also investigated the use of recent automated approaches to create those subsets (Gao, Gimpel, and Jensson 2020). However, the resulting subsets rather targeted syntactic instead of semantic similarity. One reason for this may be that approaches to generate cloze-tests consider only single-token gaps whereas the misconceptions consist of several words that form a descriptive statement.



(a) Easy Example



(b) Difficult Example

Figure 3: Example tweet from the user study with an easy misconception set (used in the study) and a difficult misconception set.

Ordering Strategy. All participants are randomly assigned to one out of four groups (ten participants per group), each corresponding to a strategy that leads to a different ordering of annotated instances. We investigate the following strategies:

Random is the control group that consists of randomly ordered instances.

AC$_{\mathrm{mlm}}$ uses the masked language modeling loss. It is a pre-computed, heuristic estimator and had (on average) the highest and most stable correlation to annotation time in our experiments with simulated annotators.

AC$_{\mathrm{GP}}$ uses a Gaussian Process that showed the highest performance on the sentence-labeling task (SPEC) in our simulated annotator experiments (cf. Table 4). It is trained interactively to predict the annotation time. We train a personalized model for each annotator using S-BERT embeddings of the presented tweet.

ACgold consists of instances explicitly ordered from very easy to very difficult using the pre-defined distractor sets. Although such annotation difficulties are unavailable in real-world annotation studies, it provides an upper-bound for the study.

Control Instances. To provide a fair comparison between different groups, we further require participants to annotate instances that quantify the difference with respect to prior knowledge and annotation proficiency. For this, we select the first ten instances and present them in the same order for all annotators. To avoid interdependency effects between the control instances and the instances used to evaluate $AC_{\{*\}}$, we selected instances that have disjoint sets of misconceptions.

Balancing Annotation Difficulty. We generate instances of different annotation difficulties using the sets of semantically similar and dissimilar misconceptions that serve as our distractors. We randomly assign an equal number of tweet-misconception pairs to each difficulty-level ranging from very easy to very difficult. The resulting 50 instances for our final study span similar ranges in terms of length as shown in Table 5 which is crucial to minimize the influence of reading time on our results. Overall, each of the five difficulty-levels consists of ten (two for the control instances) unique tweets that are annotated by all participants in different order.

# Chars	very easy	easy	medium	difficult	very difficult
T	219	211	183	217	194
T & MC	638	603	599	586	593

Table 5: Average number of characters per tweet (T) and tweet and misconception (T & MC) across all difficulty-levels of annotated items.

Study Process. The final study consists of 50 instances that are ordered corresponding to the group a participant has been assigned to. Each instance consists of a tweet and six possible misconceptions (one expert-annotated and five distractors) from which the participants are asked to select the most appropriate one. The lists of the six presented misconceptions are ordered randomly to prevent that participants learn to annotate a specific position. Finally, we ask each participant to answer a questionnaire that measures the perceived difficulty of the annotated instances.

5.3 General Results

In total, each of the 40 participants has provided 60 annotations, resulting in 400 annotations for the ten control instances (100 per group) and 2,000 annotations for the 50 final study instances (500 per group). In terms of annotation difficulty, each of the five difficulty-levels consists of 80 annotations for the control instances and 400 annotations for the final study. To assess the validity of $AC_{\{*\}}$, we require two criteria to be fulfilled:

- **H1** The participant groups do not significantly differ in terms of annotation time or annotation quality for the control instances.
- **H2** $AC_{\{*\}}$ shows a significant difference in annotation time or annotation quality compared to Random or each other.

Outliers. Across all 2,400 annotations, we identify only two cases where participants required more than ten minutes for annotation and are apparent outliers. To avoid removing annotations for evaluation, we compute the mean and standard deviation of the annotation time across all annotations (excluding the two outliers) and set the maximum value to $t_{\max} = \mu + 5\sigma = 156.39$ seconds. This results in ten annotations that are set to $t_{\max}$ for Random, three for AC_{mlm}, one for AC_{GP}, and zero for AC_{gold}. Note that this mainly favors the random control group that serves as our baseline.

	Σ_t	μ_t	σ_t	25%	50%	75%
Random	1,852.9	27.3	27.2	12.9	18.2	29.5
AC_{mlm}	1,273.4	23.2	19.4	**11.7**	18.6	27.4
AC_{GP}	1,324.3	26.4	19.0	14.9	20.7	30.8
AC_{gold}	**1,059.6**	**21.2**	**12.8**	12.6	**18.0**	**26.5**

Table 6: Mean, standard deviation, and 25%, 50%, and 75% percentiles of annotation (in seconds). Σ_t denotes the total annotation time an annotator of the respective group required to finish the study (on average).

Annotation Time. Table 6 shows the results of the final study in terms of annotation time per group. Overall, annotators of AC_{gold} required on average the least amount of time per instance and had the lowest standard deviation. We also observe a substantial decrease in the maximum annotation time, as shown in the 75th percentile for AC_{gold}. Conducting a Kruskal–Wallis test (Kruskal and Wallis 1952) on the control instances across all participant groups results in a p-value of $p = 0.200 < 0.05$.[10] Hence, we cannot reject the null-hypothesis for the control instances, and conclude that all groups initially do not show statistically significant differences in terms of annotation time for the control instances, thereby satisfying H1. Next, we conduct the same test on the evaluation instances and observe a statistically significant p-value of $p = 4.53^{-6} < 0.05$. For a more specific comparison, we further conduct pairwise Welch's t-test (Welch 1951) for each strategy with a Bonferroni-corrected p-value of $p = \frac{0.05}{6} = 0.008\overline{3}$ to account for multiple comparisons (Bonferroni 1936). Overall, AC_{gold} performs best, satisfying H2 with statistically significant improvements over Random ($p = 7.28^{-6}$) and AC_{GP} ($p = 3.79^{-7}$). Although the difference to AC_{mlm} is substantial, it is not statistically significant ($p = 0.0502$). The best performing estimator is AC_{mlm} which performs significantly better than Random ($p = 0.0069$) and substantially better than AC_{GP} ($p = 0.0084$). Between AC_{GP} and Random, we cannot observe any statistically significant differences ($p = 0.5694$).

	μ_{acc}	σ_{acc}	25%	50%	75%
Random	84.7	4.22	82.0	**86.0**	**88.0**
AC_{mlm}	83.6	5.32	80.0	84.0	86.0
AC_{GP}	83.6	**2.95**	82.0	**86.0**	86.0
AC_{gold}	**85.6**	3.01	**84.0**	84.0	**88.0**

Table 7: Mean, standard deviation, and 25%, 50%, and 75% percentiles of annotation quality (in percent accuracy).

10 In general, ANOVA (analysis of variance) is a more expressive test that does not require pairwise comparisons that are necessary for the less expressive Kruskal–Wallis test. However, we cannot apply ANOVA in our case due to violated conditions on normality and homoscedasticity of the collected data.

Annotation Quality. We evaluate annotation quality by computing the accuracy for each participant, that is, the percentage of misconceptions that they were able to correctly identify out of the six presented ones. Table 7 shows our results in terms of accuracy. Although $\mathrm{AC}_{\mathrm{gold}}$ has the highest mean accuracy, the most differences lie within the range of 2% accuracy, which is equivalent to only a single wrongly annotated instance. Conducting Kruskal–Wallis tests for the control instances shows that the difference in terms of accuracy is not statistically significant ($p = 0.881$), satisfying H1. However, the same test shows no statistically significant difference for the final study ($p = 0.723$). One reason for this may be our decision to conduct the study with voluntary participants and their higher intrinsic motivation to focus on annotation quality over annotation time (Chau et al. 2020). In contrast to crowdsourcing scenarios where annotators are mainly motivated by monetary gain—trying to reduce the amount of time they spend on their annotation at the cost of quality—voluntary annotators are more motivated to invest additional time to provide correct annotations; even more so in a setup with a low number of 60 instances.

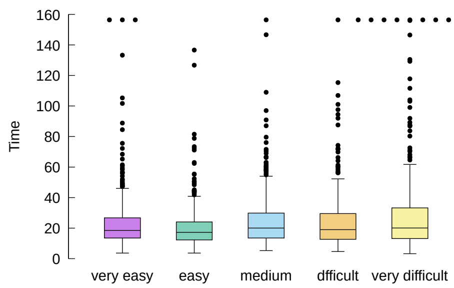


Figure 4: Annotation time (in seconds) grouped by difficulty level.

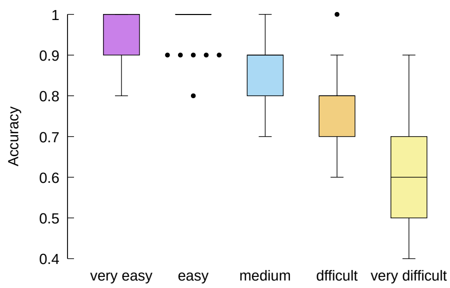


Figure 5: Accuracy per annotator grouped by difficulty level.

Difficulty Evaluation. To validate our generation approach with distractors, we further evaluate all annotation instances in terms of their annotation difficulty. As Figures 4 and 5 show, one can observe non-negligible differences in terms of annotation time as well as accuracy across instances of different difficulties. Conducting pairwise Welch's t-tests with a Bonferroni corrected p-value of $p = \frac{0.05}{10} = 0.005$ shows that in terms of accuracy, only very easy and easy instances do not express a statistically significant difference ($p = 0.25$), showing that participants had more trouble in identifying the correct misconception for difficult instances.[11] For all other instances, we observe p-values smaller than $1\mathrm{e}^{-6}$, as shown in Figure 6. In terms of annotation time, the differences are not as apparent as in annotation accuracy. We find statistically significant differences in only four out of ten cases showing that the annotation difficulty does not necessarily impact the annotation time. Overall, we still observe that instances express significant differences in terms of either annotation time or quality (or both), showing that our approach using distractor sets to control the annotation difficulty worked well.

11 Overall, we require $\frac{n(n-1)}{2}$ pairwise comparisons resulting in 10 comparisons with $n = 5$.

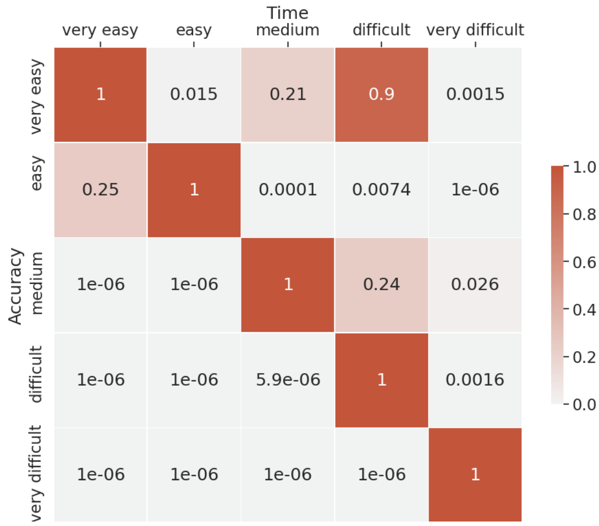


Figure 6: P-values for time (in seconds) and accuracy between different difficulty levels.

5.4 Error Analysis

	μ_t	σ_t	25%	50%	75%
MAE	12.4	6.1	8.5	10.4	14.3
RMSE	17.2	9.1	11.1	13.9	20.3
R^2	0.0	0.0	-0.1	0.0	0.0
ρ	-0.1	0.2	-0.3	-0.1	0.1

Table 8: Leave-one-out cross validation results on annotation times, grouped by user and averaged.

Model Performance. While AC_{mlm} and AC_{gold} both outperform the random baseline significantly, AC_{GP} does not. To analyze how well the used GP model performs for individual annotators, we perform leave-one-user-out cross validation experiments across all 40 participants. Table 8 shows the mean absolute error (MAE), the rooted mean squared error (RMSE), the coefficient of determination (R^2), and Spearman's ρ of our experiments. Overall, we find a low correlation between the predicted and true annotation time and high standard deviations across both errors. Further analyzing the performance of AC_{GP} for interactively predicting the annotation time (cf. Figure 7) shows that the model adapts rather slowly to additional data. As can be observed, the low performance of the model (MAE between $10 - 20$ seconds) results in a high variation in the annotation time of the selected instances between subsequent iterations; further experiments strongly suggest this is due to the model suffering from a cold start and the small amount of available training data as also discussed below.

Correlation with AC_{gold}. A second shortcoming of AC_{GP} becomes apparent when observing the difficulty of the sampled instances across all iterations, shown in Figure 8.

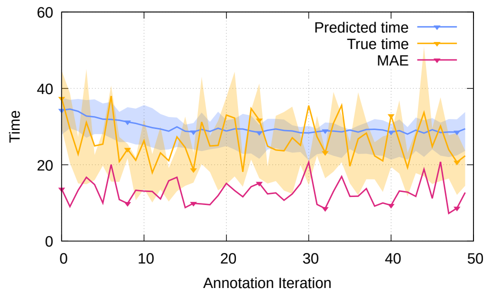


Figure 7: Mean, lower, and upper percentiles for predicted and true annotation time and the mean absolute error.

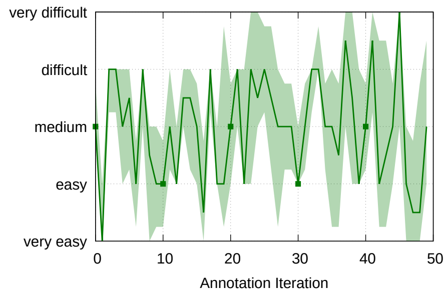


Figure 8: Median, lower, and upper percentile for instance difficulty with $\mathrm{AC}_{\mathrm{GP}}$ at each iteration.

We observe a low Spearman's ρ correlation to $\mathrm{AC}_{\mathrm{gold}}$ of 0.005, in contrast to $\mathrm{AC}_{\mathrm{mlm}}$ ($\rho = 0.22$). Only Random has a lower correlation of $\rho = -0.15$. This shows that model adaptivity plays an important role especially in low-data scenarios such as in early stages during annotation studies. We plan to tackle this issue in future work using more sophisticated models and combined approaches that initially utilize heuristics and switch to interactively trained models with the availability of sufficient training data.

5.5 Participant Questionnaire

After completing the annotation study, each participant answered a questionnaire quantifying their language proficiency, previous annotation experience, and perceived difficulty of the annotation task.

Language Proficiency. In addition to their CEFR language proficiency (Council of Europe. 2001), we further asked participants to provide optional information about their first language and the number of years they have been actively speaking English. On average, our participants have been actively speaking English for more than 10 years. Overall, they stated a language proficiency of: B1 (1), B2 (11), C1 (17), and C2 (11). Most of our participants stated German as their first language (30). Other first languages include Vietnamese (4), Chinese (3), Russian (1), and Czech (1).[12]

Annotation Experience. We further collected data from our participants regarding their previous experience as study participants as well as study conductors. In general, about 50% of our participants (18) had not participated in annotation studies before. Nineteen had participated in a few (one to three) studies, and only three in more than three studies. Even more participants had not previously conducted a study (24) or only a few (12). In total, four participants stated that they had setup more than three annotation studies.

Confounding Factors. We identify the language proficiency and previous experience with annotation studies as potential confounding factors (VanderWeele and Shpitser 2013).

12 One participant decided not to disclose any additional information except English proficiency.

	CEFR		Annotator		Conductor	
	ρ	p-value	ρ	p-value	ρ	p-value
Time	-0.307	0.054	-0.134	0.409	0.085	0.600
Accuracy	0.319	0.044	-0.060	0.711	-0.211	0.191

Table 9: Spearman's ρ correlation analysis for three potential confounding factors.

Confounding factors are variables that are difficult to control for, but have an influence on the whole study and can lead to a misinterpretation of the results. Especially in studies that include a randomized setup such as in ours—due to the random assignment of our participants into the four groups—it is crucial to investigate the influence of potential confounding factors. In our analysis, we focus on variables for which all participants provided an answer, namely, their CEFR level and their experience as participants in and conductors of annotation studies (some of our participants were researchers). Table 9 shows the results of a Spearman's ρ correlation analysis for all three variables against annotation time and accuracy. As we can see, the participants' experiences as annotators (Annotator) or study conductors (Conductor) only yields a low, non-significant correlation with time and accuracy and consequently, can be excluded as confounding factors. The influence of their language proficiency (CEFR) is more interesting, as it shows a small negative correlation for annotation time and a small positive correlation for annotation accuracy with p-values around 0.05, meaning that participants with a lower CEFR level required less time, but also had a lower accuracy. To investigate the influence of a participant's language proficiency on our results, we conduct a Kruskal–Wallis test for the distribution of different language proficiency levels across the four groups and find that they do not differ significantly with a p-value of $p = 0.961$. Nonetheless, we find that the CEFR level is an important confounding factor that needs to be considered in future study setups.

Perceived Difficulty. To quantify if there exists any difference between the actual difficulty and the perceived difficulty, we further asked our participants the following questions:

PQ1: How difficult did you find the overall annotation task?
PQ2: Did you notice any differences in difficulty between individual tweets?
PQ3: Would you have preferred a different ordering of the items?

Figure 9 shows the distribution of answers (from very easy to very difficult) to PQ1 across all four groups. Interestingly, whereas participants of the $\mathrm{AC}_{\mathrm{mlm}}$ group did require less time during their annotation compared to $\mathrm{AC}_{\mathrm{GP}}$, more people rated the study as of medium difficulty than participants of $\mathrm{AC}_{\mathrm{GP}}$. This may be an indicator that $\mathrm{AC}_{\mathrm{GP}}$ may—although not measurable in terms of annotation time—alleviate the perceived difficulty for participants, hence, still reducing the cognitive burden. We will investigate this in further studies that also include an item specific difficulty annotation, that is, by explicitly asking annotators for the perceived difficulty.[13] Overall, only four out of 40 participants (two for $\mathrm{AC}_{\mathrm{GP}}$ and one for $\mathrm{AC}_{\mathrm{mlm}}$ and $\mathrm{AC}_{\mathrm{gold}}$ each) did state to not have noticed any differences in terms of difficulty between different instances; showing that the selected distractors resulted in instances of noticeably different annotation difficulty

13 We excluded this additional annotation in the study as one pass already required $\sim 45 - 60$ minutes.

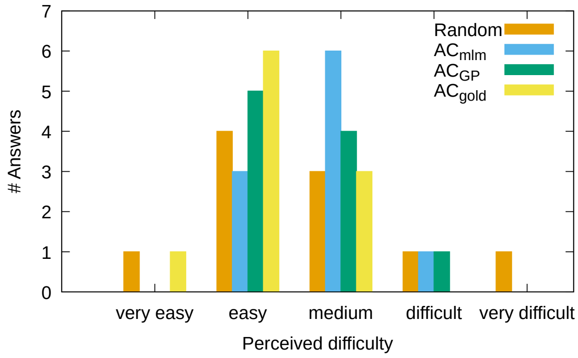


Figure 9: Accumulated perceived difficulty answers across all groups.

(PQ2). For PQ3, we find that 33 participants did not wish for a different ordering of instances (but were still allowed to provide suggestions), four would have preferred an "easy-first", one a "difficult-first", and two an entirely different ordering strategy. From the 14 free-text answers and feedback via other channels, we identify three general suggestions that may be interesting for future research:

S1: Grouping by word rarity.
S2: Grouping instances by token overlap.
S3: Grouping instances by topic (tweet or alternatively, misconception) similarity.

Further analyzing the free-text answers together with the pre-defined answers ("no", "easy-first", "difficult-first", and "other") shows that the participants disagree on the preferred ordering strategy. For instance, the participants that suggested S3, disagreed if instances should be grouped by topic similarity to reduce the number of context switches or be as diverse as possible to provide some variety during annotation. Another five participants (two from Random and one from the other groups each) even explicitly supported a random ordering in the free-text answer. The disagreement upon the ordering strategy shows the importance of interactively trained estimators that are capable of providing personalized annotation curricula.

6. Limitations and Future Work

We evaluated AC with an easy-instances-first strategy in simulations as well as in a highly controlled setup using a finite, pre-annotated data set and task-agnostic estimators to minimize possible noise factors. To demonstrate the viability of AC with a sufficient number of voluntary annotators, we further chose a dataset that covers a widely discussed topic and manually controlled the annotation difficulty to make it accessible for non-experts. To evaluate AC with more generalizable results in a real-world scenario, we discuss existing limitations that should be considered beforehand that can also serve as promising research directions for future work.

Difficulty Estimators. Due to novelty of the proposed approach and the lack of well-established baselines, we focused on task-agnostic annotation difficulty estimators such as reading difficulty and annotation time, which can easily be applied to a wide range of tasks. Although our study results show that they work to some extent, our evaluation with existing datasets also shows that especially non-adaptive estimators, which approximate the absolute task-difficulty, are sensitive to the data domain and annotation task (cf. the low performance of length-based estimators on the SigIE data in Section 4). Such issues could be addressed by implementing estimators that are more *task-specific*. For named entity annotations, a general improvement may be achieved by considering the number of nouns within a sentence that can be obtained from a pre-trained part-of-speech tagger. One may even consider domain-specific word frequency lists to provide a difficulty estimate for entities. For instance, among the annotated named entities in Muc7_T "U.S." (occurs 72 times) may be easier to annotate than "Morningstar" (occurs only once); simply based on a word frequency analysis. Other, more sophisticated approaches from educational research such as item response theory (Baker 2001) and scaffolding (Jackson et al. 2020) may also lead to better task-agnostic estimators. Such approaches and combinations of task-agnostic with task-specific estimators remain to be investigated in future work.

Annotation Strategies. In this work, we focused on developing and evaluating a strategy for our non-expert annotation scenario. Although it proved to be effective in our user study, we also find that our annotators disagree in their preferences with respect to the ordering of instances—which indicates that investigating *annotator-specific* strategies could be a promising line for future work. Another shortcoming of the evaluated strategy is that it does not consider an annotator's boredom or frustration (Vygotsky 1978). Especially when considering larger annotation studies, motivation may become an increasingly important factor with non-expert annotators as they further progress in a task and become more proficient. Such a strategy may also be better suited for annotation scenarios that involve domain experts to retain a high motivation by avoiding boredom—for instance, by presenting them with subsequent instances of varying difficulty or different topics. Domain experts who do not require a task-specific training may also benefit from strategies that focus on familiarizing them with the data domain early on to provide them with a good idea of what kind of instances they can expect throughout their annotations. To implement strategies that consider annotator-specific factors such as motivation and perceived difficulty, adaptive estimators may have an advantage over non-adaptive ones as they can incorporate an annotator's preference on the fly. We will investigate more sophisticated adaptive estimators (also coupled with non-adaptive ones) and strategies in future work and also plan to evaluate AC with domain expert annotators.

Larger Datasets. While using a finite set of annotated instances was necessary in our user study to ensure a proper comparability, AC is not limited to annotation scenarios with finite sets. However, deploying AC in scenarios that involve a large number of unlabeled instances requires additional consideration besides an annotator's motivation. In scenarios that only annotate a subset of the unlabeled data (similar to pool-based active learning), an easy-instances-first strategy may lead to a dataset that is imbalanced toward instances that are easy to annotate. This can hurt data diversity and consequently result in models that do not generalize well to more difficult instances. To create more diverse datasets, one may consider introducing a stopping criterion (e.g., a fixed threshold) for the annotator training phase and moving on to a different sampling strategy from active learning. Other, more sophisticated approaches would be to utilize adaptive estimators with a pacing function (Kumar, Packer, and Koller 2010) or sampling objectives that jointly consider annotator training and data diversity (Lee, Meyer, and Gurevych 2020). Such approaches are capable of monitoring the study progress and can react accordingly, which may result in more diverse datasets. However, they also face additional limitations in terms of the computational overhead that may require researchers to consider an asynchronous model training in their setup.

Implementation Overhead. Finally, to apply AC in real-world annotation studies, one needs to consider the additional effort for study conductors to implement it. Whereas the task-agnostic estimators we provide can be integrated with minimal effort, developing task- and annotator-specific estimators may not be a trivial task and requires a profound knowledge about the task, data, and annotators. Another open question is how well the time saving of approximately 8–13 minutes per annotator in our study translates to large-scale annotation studies. If so, then AC could also be helpful in annotation studies with domain experts by resulting in more annotated instances within a fixed amount of time—however, if not, this would simply lead to a trade-off between the time investment of the study conductor and annotators. Overall, we find that developing and evaluating further strategies and estimators to provide study conductors with a wide range of choices to consider for their annotation study will be an interesting task for the research community.

7. Conclusion

With annotation curricula, we have introduced a novel approach for implicitly training annotators. We provided a formalization for an easy-instances-first strategy that orders instances from easy to difficult by approximating the annotation difficulty with task-agnostic heuristics and annotation time. In our experiments with three English datasets, we identified well-performing heuristics and interactively trained models and find that the data domain and the annotation task can play an important role when creating an annotation curriculum. Finally, we evaluate the best performing heuristic and adaptive model in a user study with 40 voluntary participants who classified English tweets about the Covid-19 pandemic and show that leveraging AC can lead to a significant reduction in annotation time while preserving annotation quality.

With respect to our initial research questions (cf. Section 1), our results show that the order in which instances are annotated can have a statistically significant impact in terms of annotation time (RQ1) and that recent language models can provide a strong baseline to pre-compute a well-performing ordering (RQ2). We further find that our interactively trained regression models lack adaptivity (RQ3), as they perform well on existing datasets with hundreds or more training instances, but fall behind non-adaptive estimators in the user study.

We conclude that annotation curricula provide a promising way for more efficient data acquisition in various annotation scenarios—but that they also need further investigation with respect to task-specific estimators for annotation difficulty, annotator-specific preferences, and applicability on larger datasets. Our analysis of existing work shows that, unfortunately, the annotation ordering as well as annotation times are seldomly reported. In the face of the increasing use of AI models in high-stake domains (Sambasivan et al. 2021) and the potentially harmful impact of biased data (Papakyriakopoulos et al. 2020), we ask dataset creators to consider including individual annotation times and orderings along with a datasheet (Gebru et al. 2021) when publishing their dataset. To facilitate future research, we share all code and data and provide a ready-to-use and extensible implementation of AC in the INCEpTION annotation platform.[14]

14 https://inception-project.github.io/

Acknowledgments
This work has been supported by the European Regional Development Fund (ERDF) and the Hessian State Chancellery – Hessian Minister of Digital Strategy and Development under the promotional reference 20005482 (TexPrax) and the German Research Foundation under grant № EC 503/1-1 and GU 798/21-1 (INCEpTION). We thank Michael Bugert, Richard Eckart de Castilho, Max Glockner, Ulf Hamster, Yevgeniy Puzikov, Kevin Stowe, and the anonymous reviewers for their thoughtful comments and feedback, as well as all anonymous participants in our user study.

Ethics Statement

Informed consent. Participants of our user study participated voluntarily and anonymously with a self-chosen, unique study key that allows them to request the deletion of their provided data at a later point. Upon registration, they are informed about the data presented and collected in the study, its further use, and the purpose of the study. Before collecting any data, participants are explicitly asked for their informed consent. We do not collect any personal data in our study. If participants do not provide their informed consent, their study key is deleted immediately. For publication, the study key is further replaced with a randomly generated user id.

Use of Twitter data. The CovidLies corpus (Hossain et al. 2020) we used to generate the instances for our annotation study consists of annotated tweets. To protect the anonymity of the user who created the tweet, we only display the text (removing any links) without any metadata like Twitter user id, or timestamps to our study participants. We only publish the tweet ids in our study data to conform with Twitter's terms of service and hence, all users retain their right to delete their data at any point.

References

Agarwal, Manish and Prashanth Mannem. 2011. Automatic Gap-fill Question Generation from Text Books. In *Proceedings of the Sixth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 56–64, Portland, Oregon, USA.

Ash, Jordan T., Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. 2020. Deep Batch Active Learning by Diverse, Uncertain Gradient Lower Bounds. In *International Conference on Learning Representations*, pages 1–26, Online.

Baker, Frank. 2001. *The basics of item response theory*. ERIC Clearinghouse on Assessment and Evaluation, College Park, Md.

Beck, Tilman, Ji-Ung Lee, Christina Viehmann, Marcus Maurer, Oliver Quiring, and Iryna Gurevych. 2021. Investigating label suggestions for opinion mining in German covid-19 social media. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–13, Online.

Beigman Klebanov, Beata and Eyal Beigman. 2014. Difficult cases: From data to learning, and back. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 390–396, Baltimore, Maryland.

Beinborn, Lisa, Torsten Zesch, and Iryna Gurevych. 2014. Predicting the Difficulty of Language Proficiency Tests. *Transactions of the Association for Computational Linguistics*, 2:517–529.

Bengio, Yoshua, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference On Machine Learning*, pages 41–48, Montreal, Canada.

Bonferroni, Carlo. 1936. Teoria statistica delle classi e calcolo delle probabilita. *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commericiali di Firenze*, 8:3–62.

Bowman, Samuel R., Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal.

Chau, Hung, Saeid Balaneshin, Kai Liu, and Ondrej Linda. 2020. Understanding the Tradeoff between Cost and Quality of Expert Annotations for Keyphrase Extraction. In *Proceedings of the 14th Linguistic Annotation Workshop*, pages 74–86, Barcelona, Spain.

Council of Europe. 2001. *Common European Framework of Reference for Languages: learning, teaching, assessment*. Council for Cultural Co-operation. Education Committee. Modern Languages Division. Cambridge University Press, Strasbourg, France.

Deutsch, Tovly, Masoud Jasbi, and Stuart Shieber. 2020. "linguistic features for readability assessment". In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 1–17, Online.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, USA.

Fang, Meng, Jie Yin, and Dacheng Tao. 2014. Active Learning for Crowdsourcing Using Knowledge Transfer. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1809–1815, Québec City, Québec, Canada.

Fang, Meng, Xingquan Zhu, Bin Li, Wei Ding, and Xindong Wu. 2012. Self-taught active learning from crowds. In *2012 IEEE 12th International Conference on Data Mining*, pages 858–863, Brussels, Belgium.

Felice, Mariano and Paula Buttery. 2019. Entropy as a Proxy for Gap Complexity in Open Cloze Tests. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 323–327, Varna, Bulgaria.

Fort, Karën and Benoît Sagot. 2010. Influence of Pre-Annotation on POS-Tagged Corpus Development. In *Proceedings of the Fourth Linguistic Annotation Workshop*, pages 56–63, Uppsala, Sweden.

Gal, Yarin, Riashat Islam, and Zoubin Ghahramani. 2017. Deep Bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1923–1932, Sydney,

Australia.
Gao, Lingyu, Kevin Gimpel, and Arnar Jensson. 2020. Distractor Analysis and Selection for Multiple-Choice Cloze Questions for Second-Language Learners. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 102–114, Seattle, Washington, USA.
Gebru, Timnit, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III, and Kate Crawford. 2021. Datasheets for datasets. *Commun. ACM*, 64(12):86–92.
Geva, Mor, Yoav Goldberg, and Jonathan Berant. 2019. Are We Modeling the Task or the Annotator? An Investigation of Annotator Bias in Natural Language Understanding Datasets. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1161–1166, Hong Kong, China.
Gururangan, Suchin, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't Stop Pretraining: Adapt Language Models to Domains and Tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online.
Hossain, Tamanna, Robert L. Logan IV, Arjuna Ugarte, Yoshitomo Matsubara, Sean Young, and Sameer Singh. 2020. COVIDLies: Detecting COVID-19 misinformation on social media. In *Proceedings of the 1st Workshop on NLP for COVID-19 (Part 2) at EMNLP 2020*, pages 1–11, Online.
Huang, Sheng-Jun, Rong Jin, and Zhi-Hua Zhou. 2010. Active Learning by Querying Informative and Representative Examples. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 1*, pages 892—900, Vancouver, Canada.
Jackson, Corey, Carsten Østerlund, Kevin Crowston, Mahboobeh Harandi, Sarah Allen, Sara Bahaadini, Scotty Coughlin, Vicky Kalogera, Aggelos Katsaggelos, Shane Larson, et al. 2020. Teaching citizen scientists to categorize glitches using machine learning guided training. *Computers in Human Behavior*, 105:1–11.
Ke, Guolin, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems*, pages 3149–3157.
Kelly, A. V. 2009. *The Curriculum : Theory and Practice*. SAGE Publications, Thousand Oaks, CA.
Kicikoglu, Osman Doruk, Richard Bartle, Jon Chamberlain, Silviu Paun, and Massimo Poesio. 2020. Aggregation Driven Progression System for GWAPs. In *Workshop on Games and Natural Language Processing*, pages 79–84, Marseille, France.
Kincaid, J Peter, Robert P Fishburne Jr, Richard L Rogers, and Brad S Chissom. 1975. Derivation Of New Readability Formulas (Automated Readability Index, Fog Count And Flesch Reading Ease Formula) For Navy Enlisted Personnel. Technical report, Naval Technical Training Command Millington TN Research Branch.
Kirsch, Andreas, Joost van Amersfoort, and Yarin Gal. 2019. BatchBALD: Efficient and Diverse Batch Acquisition for Deep Bayesian Active Learning. In *Advances in Neural Information Processing Systems*, pages 7026–7037, Vancouver, Canada.
Klie, Jan-Christoph, Michael Bugert, Beto Boullosa, Richard Eckart de Castilho, and Iryna Gurevych. 2018. The INCEpTION Platform: Machine-Assisted and Knowledge-Oriented Interactive Annotation. In *Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations*, pages 5–9, Association for Computational Linguistics.
Klie, Jan-Christoph, Richard Eckart de Castilho, and Iryna Gurevych. 2020. From Zero to Hero: Human-In-The-Loop Entity Linking in Low Resource Domains. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6982–6993, Online.
Krashen, Stephen. 1982. *Principles and Practice in Second Language Acquisition*. Pergamon Press, Oxford New York.
Kruskal, William H. and W. Allen Wallis. 1952. Use of Ranks in One-Criterion Variance Analysis. *Journal of the American Statistical Association*, 47(260):583–621.
Kumar, M., Benjamin Packer, and Daphne Koller. 2010. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, volume 23, pages 1189–1197.
Laws, Florian, Christian Scheible, and Hinrich Schütze. 2011. Active Learning with Amazon Mechanical Turk. In

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1546–1556, Edinburgh, Scotland, UK.

Lee, Ji-Ung, Christian M. Meyer, and Iryna Gurevych. 2020. Empowering Active Learning to Jointly Optimize System and User Demands. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4233–4247, Online.

Lee, Ji-Ung, Erik Schwan, and Christian M. Meyer. 2019. Manipulating the Difficulty of C-Tests. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 360–370, Florence, Italy.

Lewis, David D. and William A. Gale. 1994. A Sequential Algorithm for Training Text Classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 3–12, Dublin, Ireland.

Lingren, Todd, Louise Deleger, Katalin Molnar, Haijun Zhai, Jareen Meinzen-Derr, Megan Kaiser, Laura Stoutenborough, Qi Li, and Imre Solti. 2014. Evaluating the impact of pre-annotation on annotation speed and potential bias: natural language processing gold standard development for clinical named entity recognition in clinical trial announcements. *Journal of the American Medical Informatics Association*, 21(3):406–413.

Loukina, Anastassia, Su-Youn Yoon, Jennifer Sakano, Youhua Wei, and Kathy Sheehan. 2016. Textual complexity as a predictor of difficulty of listening items in language proficiency tests. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3245–3253, Osaka, Japan.

Lowell, David, Zachary C. Lipton, and Byron C. Wallace. 2019. Practical Obstacles to Deploying Active Learning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 21–30, Hong Kong, China.

Madge, Chris, Juntao Yu, Jon Chamberlain, Udo Kruschwitz, Silviu Paun, and Massimo Poesio. 2019. Progression in a Language Annotation Game with a Purpose. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, pages 77–85, Honolulu, Hawaii, USA.

Martínez Alonso, Héctor, Barbara Plank, Anders Johannsen, and Anders Søgaard. 2015. Active learning for sense annotation. In *Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015)*, pages 245–249, Vilnius, Lithuania.

Mayfield, Elijah and Alan W Black. 2020. Should You Fine-Tune BERT for Automated Essay Scoring? In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 151–162, Online.

Mostow, Jack and Hyeju Jang. 2012. Generating Diagnostic Multiple Choice Comprehension Cloze Questions. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 136–146, Montréal, Canada.

Nguyen, Hieu T. and Arnold Smeulders. 2004. Active Learning Using Pre-Clustering. In *Proceedings of the Twenty-First International Conference on Machine Learning*, pages 79–87, Banff, Alberta, Canada.

Nie, Yixin, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A New Benchmark for Natural Language Understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4885–4901, Online.

Pak, Alexander and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, pages 1320–1326, European Language Resources Association (ELRA), Valletta, Malta.

Papakyriakopoulos, Orestis, Simon Hegelich, Juan Carlos Medina Serrano, and Fabienne Marco. 2020. Bias in word embeddings. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, page 446–457, New York, NY, USA.

Paun, Silviu, Bob Carpenter, Jon Chamberlain, Dirk Hovy, Udo Kruschwitz, and Massimo Poesio. 2018. Comparing Bayesian models of annotation. *Transactions of the Association for Computational Linguistics*, 6:571–585.

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-Learn:

Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825—-2830.

Peters, Matthew E., Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, USA.

Peters, Matthew E., Sebastian Ruder, and Noah A. Smith. 2019. To Tune or Not to Tune? Adapting Pretrained Representations to Diverse Tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 7–14, Florence, Italy.

Reimers, Nils and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China.

Rogers, Anna. 2021. Changing the world by changing the data. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2182–2194, Online.

Roy, Nicholas and Andrew McCallum. 2001. Toward Optimal Active Learning through Sampling Estimation of Error Reduction. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 441–448, Williamstown, MA, USA.

Sakaguchi, Keisuke, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106.

Salazar, Julian, Davis Liang, Toan Q. Nguyen, and Katrin Kirchhoff. 2020. Masked Language Model Scoring. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2699–2712, Online.

Sambasivan, Nithya, Shivani Kapania, Hannah Highfill, Diana Akrong, Praveen Paritosh, and Lora M Aroyo. 2021. "Everyone Wants to Do the Model Work, Not the Data Work": Data Cascades in High-Stakes AI. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, pages 1–15, Association for Computing Machinery, New York, NY, USA.

Sanh, Victor, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

Schulz, Claudia, Christian M. Meyer, Jan Kiesewetter, Michael Sailer, Elisabeth Bauer, Martin R. Fischer, Frank Fischer, and Iryna Gurevych. 2019. Analysis of Automatic Annotation Suggestions for Hard Discourse-Level Tasks in Expert Domains. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2761–2772, Florence, Italy.

Sennrich, Rico, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany.

Settles, Burr. 2012. Active Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114.

Settles, Burr, Mark Craven, and Lewis Friedland. 2008. Active Learning with Real Annotation Costs. In *Proceedings of the NIPS Workshop on Cost-Sensitive Learning*, pages 1–10, Vancouver, Canada.

Siddhant, Aditya and Zachary C. Lipton. 2018. Deep Bayesian Active Learning for Natural Language Processing: Results of a Large-Scale Empirical Study. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2904–2909, Brussels, Belgium.

Snow, Rion, Brendan O'Connor, Daniel Jurafsky, and Andrew Ng. 2008. Cheap and Fast – But is it Good? Evaluating Non-Expert Annotations for Natural Language Tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 254–263, Honolulu, Hawaii, USA.

Stab, Christian, Johannes Daxenberger, Chris Stahlhut, Tristan Miller, Benjamin Schiller, Christopher Tauchmann, Steffen Eger, and Iryna Gurevych. 2018. ArgumenText: Searching for arguments in heterogeneous sources. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 21–25, New Orleans,

Louisiana.
Sweetser, Penelope and Peta Wyeth. 2005. GameFlow: A Model for Evaluating Player Enjoyment in Games. *Computers in Entertainment*, 3(3):3.
Szpiro, George. 2010. *Numbers rule : the vexing mathematics of democracy, from Plato to the present*. Princeton University Press.
Tauchmann, Christopher, Johannes Daxenberger, and Margot Mieskes. 2020. The Influence of Input Data Complexity on Crowdsourcing Quality. In *Proceedings of the 25th International Conference on Intelligent User Interfaces Companion*, pages 71—72, Cagliari, Italy.
Taylor, Wilson L. 1953. Cloze Procedure": A New Tool for Measuring Readability. *Journalism & Mass Communication Quarterly*, 30(4):415–433.
Tomanek, Katrin and Udo Hahn. 2009. Timed Annotations - Enhancing MUC7 Metadata by the Time It Takes to Annotate Named Entities. In *Proceedings of the Third Linguistic Annotation Workshop*, pages 112–115, Singapore.
Turner, Brandon M. and Dan R. Schley. 2016. The anchor integration model: A descriptive model of anchoring effects. *Cognitive Psychology*, 90:1–47.
VanderWeele, Tyler J and Ilya Shpitser. 2013. On the definition of a confounder. *Annals of statistics*, 41(1):196–220.
Vygotsky, Lev. 1978. *Mind in society: The development of higher psychological processes*. Harvard University Press.
Wang, Min, Fan Min, Zhi-Heng Zhang, and Yan-Xue Wu. 2017. Active learning through density clustering. *Expert Systems with Applications*, 85:305–317.
Welch, Bernard Lewis. 1951. On the Comparison of Several Mean Values: An Alternative Approach. *Biometrika*, 38(3/4):330–336.
Xia, Menglin, Ekaterina Kochmar, and Ted Briscoe. 2016. "text readability assessment for second language learners". In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 12–22, San Diego, CA, USA.
Yang, Yinfei, Oshin Agarwal, Chris Tar, Byron C. Wallace, and Ani Nenkova. 2019. Predicting Annotation Difficulty to Improve Task Routing and Model Performance for Biomedical Information Extraction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1471–1480, Minneapolis, Minnesota, USA.
Yeung, Chak Yan, John Lee, and Benjamin Tsou. 2019. Difficulty-aware Distractor Generation for Gap-Fill Items. In *Proceedings of the The 17th Annual Workshop of the Australasian Language Technology Association*, pages 159–164, Sydney, Australia.
Yimam, Seid Muhie, Chris Biemann, Richard Eckart de Castilho, and Iryna Gurevych. 2014. Automatic Annotation Suggestions and Custom Annotation Layers in WebAnno. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 91–96, Baltimore, Maryland, USA.
Yuan, Michelle, Hsuan-Tien Lin, and Jordan Boyd-Graber. 2020. Cold-start Active Learning through Self-supervised Language Modeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7935–7948, Online.
Zhang, Chicheng and Kamalika Chaudhuri. 2015. Active Learning from Weak and Strong Labelers. In *Advances in Neural Information Processing Systems*, pages 703–711, Montréal, Canada.
Zhu, Jingbo, Huizhen Wang, Tianshun Yao, and Benjamin K Tsou. 2008. Active Learning with Sampling by Uncertainty and Density for Word Sense Disambiguation and Text Classification. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 1137–1144, Manchester, UK.

Erratum: Annotation Curricula to Implicitly Train Non-Expert Annotators

Ji-Ung Lee*
UKP Lab / TU Darmstadt

Jan-Christoph Klie*
UKP Lab / TU Darmstadt

Iryna Gurevych
UKP Lab / TU Darmstadt

The authors of this work discovered an incorrect inequality symbol in section 5.3 (page 360). The paper stated that the differences in the annotation times for the control instances result in a p-value of 0.200 which is smaller than 0.05 ($p = 0.200 < 0.05$). As 0.200 is of course larger than 0.05, the correct inequality symbol is $p = 0.200 > 0.05$, which is in line with the conclusion that follows in the text. The paper has been updated accordingly.

* Equal contribution

Part III

Epilogue

Chapter 12

Conclusion And Future Work

12.1 Conclusion

While globalization and immigration increasingly drive the necessity of second language acquisition, there still exists a large disparity in terms of methods that can alleviate the work of teachers and advance self-directed learning. In this thesis, we have devised multiple methods to alleviate existing gaps with respect to exercise generation and selection on the use case of C-Tests.

In Chapter 6, we have devised new methods that are capable of generating C-Tests with varying gap size or gap placement. To make these approaches feasible, we have further performed careful ablation experiments that resulted in six features which we used to train efficient models. Finally, in a user study with 60 participants and 16 C-Tests, we show that both strategies succeed in generating C-Tests that are significantly better.

In subsequent work in Chapter 7, we further addressed existing shortcomings of our generation strategies; i.e., their incapability to consider the whole space of possible C-Tests and their inability to adhere to hard constraints that may be posed by a teacher. We propose to address the C-Test generation problem as a mixed-integer programming problem and devise a method that addresses both shortcomings. In a user study with 40 participants and 32 C-Tests, we show that our approach significantly outperforms two of our baseline approaches, one of which is GPT-4 (OpenAI, 2023) and performs similar to the third.

In Chapter 8, we surveyed the current state of methods that aim to improve efficiency in NLP. We especially focused on data efficiency, providing an overview on existing approaches related to filtering, active learning, and curriculum learning; highlighting advantages and shortcomings. Finally, we discussed open challenges and provided pointers for future research.

In Chapter 9, we have addressed the problem of adaptive exercise selection. Taking the lack of learner-annotated data as our motivation, we proposed to utilize active learning to adaptively train a selection model. To make active learning suitable for educational scenarios—i.e., to prevent hurting the learner's progress—we formalized user and model objective (i.e., the active learning objective) and proposed sampling strategies to select C-Tests that jointly optimize both. In our experiments with simulated learners that behave according to four different patterns across five different proficiency levels, we have shown that our proposed sampling strategies yield the best results.

To transfer our joint optimization approach to other use cases, we then implemented a novel, interactive data collection system in Chapter 10. We have shown that the messaging system which is accompanied by a chatbot alleviates the documentation of errors, their causes, and their solution during work. In three studies with factory workers, we have showcased that the system can be used to collect data on-the-fly and that label suggestions are helpful—but that appropriate correction mechanisms are equally important.

In Chapter 11, we have investigated if there exists a learning process in annotation studies. For this, we first proposed a formal framework to apply different ordering strategies during annotation which we coined annotation curricula. We conducted experiments with existing datasets to identify models that are well suited to rank instances according to their annotation time. We then carefully designed a novel annotation task based on an existing dataset and controlled the annotation difficulty by generating according distractor sets, taking inspiration from cloze tests. In our user study, we found that ordering instances according to their annotation difficulty can significantly decrease the annotation time.

12.2 Future Work

Recent advances in NLP have led to the development of increasingly large models (cf. Figure 12.1). Although such LLMs have shown astonishing performances across a plethora of tasks (Gao et al., 2023), the growing need for computational resources that are required to pre-train and fine-tune LLMs has led to a growing disparity in the research community (Lee et al., 2023). At the same time, the increasing popularity of LLMs has surfaced four increasingly critical issues that need to be addressed. First, LLMs suffer from *hallucination*—i.e., generated text that is "nonsensical, or unfaithful to the provided source input" (Ji et al., 2023)—leading them to provide factually incorrect, but convincing statements (cf. Figure 12.2).[1] Second, they are susceptible to pick up spurious patterns in the data during training which can lead them to make unfair decisions that reflect societal *biases* on gender, race, or religion (Gallegos et al., 2023). Third, they require an increasing amount of computational *resources* for training and deployment, further aggravating climate change (Schwartz et al., 2020) and excluding researchers who do not have access to large GPU clusters (Lee et al., 2023). Finally, the increasing

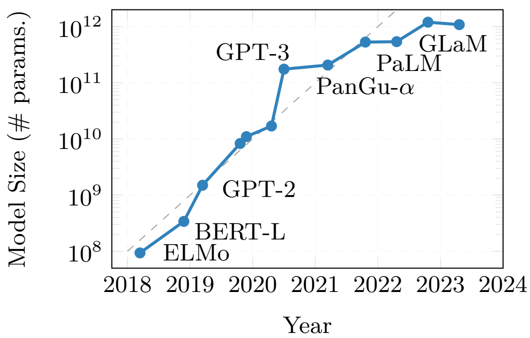


Figure 12.1: Number of parameters in pre-trained models. Adapted from Treviso et al. (2023).

[1] Date of conversation with ChatGPT (GPT-3.5): 16.01.2024
URL: https://chat.openai.com/share/6f37abd8-ab30-46bd-8b32-bf52e9bb8a42
50th Academy Awards: https://www.oscars.org/oscars/ceremonies/1978

entanglement of powerful LLMs such as GPT-4 (OpenAI, 2023) and Gemini (Anil et al., 2023) with other tools and existing infrastructure has stoked fears about their potential risks and possibilities of misuse (Wiener, 1960); raising the question of how to *align* them to human values (Russell, 2019, p. 137).

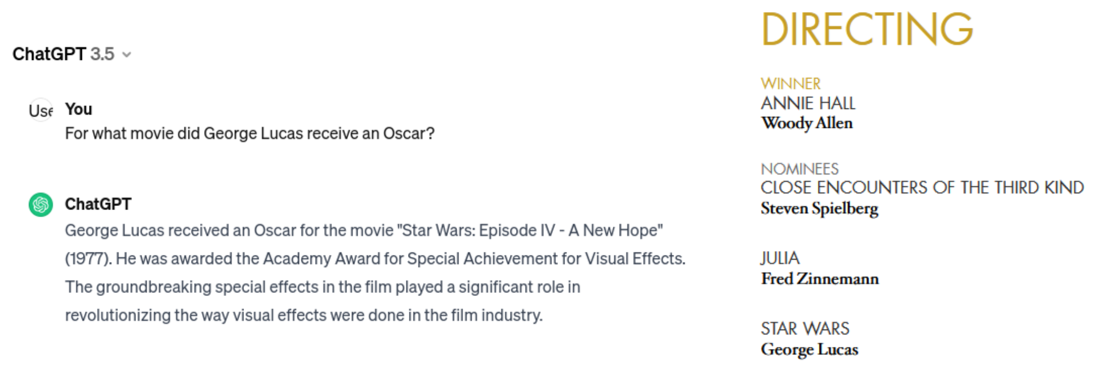


Figure 12.2: GPT-3.5 claiming that George Lucas did receive an Academy Award for the movie Star Wars (left). However, the records of 50th Academy Awards show that he was nominated but not presented with the award (right). The visual effects award referred by the model actually went to the art directors and costume designers of the movie Star Wars.

12.2.1 Interactive Machine Learning

Despite the advent of in-context learning (ICL; Radford et al. 2019), there exist clear limitations on how much LLMs can "learn" only from the input (Liu et al., 2022, 2023). Interactively adapting LLMs after pre-training is thus necessary to scale them to an arbitrary number of use cases; a paradigm also known as *interactive machine learning* (IML; Fails and Olsen 2003). In this thesis, we have devised various methods that may be investigated in future work to alleviate above issues via IML.

First, adaptivity is a key component that is required to build IML systems. Our work on joint sampling (Chapter 9) could be adapted to reduce the amount of necessary feedback and efficiently align LLMs with a user's goals and values. Our methods could further be used to alleviate existing bottlenecks in data acquisition and to improve data quality (Kreutzer et al., 2022). Second, mathematical guarantees can also improve the quality of IML systems by eliminating the need for unnecessary user interaction. Moreover, whereas such guarantees would be necessary to safely deploy LLMs in critical domains, we are currently far from achieving them. For instance, our work done in Chapter 7 may allow us to integrate trained models into constrained optimization; but we currently lack strong bounds for activation functions that go beyond ReLU (Anderson et al., 2020) and moreover, cannot scale solvers to billions of parameters. As a first step to integrate larger models into constrained optimization, we propose to utilize concept bottleneck models (Koh et al., 2020) that induce interpretable upper layers in models. These would allow us to split the model effectively into upper layers (that would be used for constrained optimization) and lower layers which would be kept frozen. The feature layer would then allow us to pose specific constraints in the input which are directly relatable to a specific feature.

12.2.2 Second Language Acquisition

Finally, we identify various open issues and challenges that need to be resolved to meet the increasing demand of second language acquisition tools.

First, there still exists a huge lack of annotated data that is often caused by data privacy or proprietary usage agreements. Despite incentives to collect more data—for instance using games with a purpose (Poesio et al., 2013; Madge et al., 2019; Kicikoglu et al., 2020) or utilizing crowdsourcing (Lyding et al., 2022)—data collection remains difficult. A first step to address the lack of data could be to build common resources that are shared across the various platforms that aim to collect educational data (Haring et al., 2021; Heck and Meurers, 2022; Chan et al., 2022; Bitew et al., 2023). Second, similar to other NLP areas, we find a huge disparity in terms of educational resources in languages other than Chinese, English, French, German, or Spanish. While extending existing incentives that target language diversity to educational data can be a starting point (Siminyu and Freshia, 2020; Salesky et al., 2023), they would require a high grade of adaptation towards language-specific properties. This brings us to the third and final issue that needs to be addressed; the development of models with sufficient multi-lingual capabilities. This is especially important as second language acquisition resources ideally should be served in a learners first language (Beinborn et al., 2014b). While multi-lingual models would not resolve all existing issues, we conjecture that they could serve as a solid foundation to scale the development of educational resources and tools to new languages.

Bibliography

Charu C. Aggarwal. 2016. *Recommender Systems*. Springer.

K. Ahmad. 1985. *Computers, Language Learning and Language Teaching*. Cambridge Language Teaching Library. Cambridge University Press.

Mary Ainley. 2006. Connecting with learning: Motivation, affect and cognition in interest processes. *Educational Psychology Review*, 18:391–405.

Samah AlKhuzaey, Floriana Grasso, Terry R Payne, and Valentina Tamma. 2023. Text-based question difficulty prediction: A systematic review of automatic approaches. *International Journal of Artificial Intelligence in Education*, pages 1–53.

Saleema Amershi, Maya Cakmak, William Bradley Knox, and Todd Kulesza. 2014. Power to the people: The role of humans in interactive machine learning. *AI Magazine*, 35(4):105–120.

Ross Anderson, Joey Huchette, Will Ma, Christian Tjandraatmadja, and Juan Pablo Vielma. 2020. Strong mixed-integer programming formulations for trained neural networks. *Mathematical Programming*, 183(1-2):3–39.

Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Slav Petrov, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy Lillicrap, Angeliki Lazaridou, Orhan Firat, James Molloy, Michael Isard, Paul R. Barham, Tom Hennigan, Benjamin Lee, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, Ryan Doherty, Eli Collins, Clemens Meyer, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha Goel, George Tucker, Enrique Piqueras, Maxim Krikun, Iain Barr, Nikolay Savinov, Ivo Danihelka, Becca Roelofs, Anaïs White, Anders Andreassen, Tamara von Glehn, Lakshman Yagati, Mehran Kazemi, Lucas Gonzalez, Misha Khalman, Jakub Sygnowski, Alexandre Frechette, Charlotte Smith, Laura Culp, Lev Proleev, Yi Luan, Xi Chen, James Lottes, Nathan Schucher, Federico Lebron, Alban Rrustemi, Natalie Clay, Phil Crone, Tomas Kocisky, Jeffrey Zhao, Bartek Perz, Dian Yu, Heidi Howard, Adam Bloniarz, Jack W. Rae, Han Lu, Laurent Sifre, Marcello Maggioni, Fred Alcober, Dan Garrette, Megan Barnes, Shantanu Thakoor, Jacob Austin, Gabriel Barth-Maron, William Wong, Rishabh Joshi, Rahma Chaabouni, Deeni Fatiha, Arun Ahuja, Ruibo Liu, Yunxuan Li, Sarah Cogan, Jeremy Chen, Chao Jia, Chenjie Gu, Qiao Zhang, Jordan Grimstad, Ale Jakse Hartman, Martin Chadwick, Gaurav Singh Tomar, Xavier Garcia, Evan Senter, Emanuel Taropa, Thanumalayan Sankaranarayana Pillai, Jacob Devlin, Michael Laskin, Diego de Las Casas, Dasha Valter, Connie Tao, Lorenzo Blanco, Adrià Puigdomènech Badia, David Reitter, Mianna Chen, Jenny Brennan, Clara Rivera, Sergey Brin, Shariq Iqbal, Gabriela Surita, Jane Labanowski, Abhi Rao, Stephanie Winkler, Emilio Parisotto,

Yiming Gu, Kate Olszewska, Yujing Zhang, Ravi Addanki, Antoine Miech, Annie Louis, Laurent El Shafey, Denis Teplyashin, Geoff Brown, Elliot Catt, Nithya Attaluri, Jan Balaguer, Jackie Xiang, Pidong Wang, Zoe Ashwood, Anton Briukhov, Albert Webson, Sanjay Ganapathy, Smit Sanghavi, Ajay Kannan, Ming-Wei Chang, Axel Stjerngren, Josip Djolonga, Yuting Sun, Ankur Bapna, Matthew Aitchison, Pedram Pejman, Henryk Michalewski, Tianhe Yu, Cindy Wang, Juliette Love, Junwhan Ahn, Dawn Bloxwich, Kehang Han, Peter Humphreys, Thibault Sellam, James Bradbury, Varun Godbole, Sina Samangooei, Bogdan Damoc, Alex Kaskasoli, Sébastien M. R. Arnold, Vijay Vasudevan, Shubham Agrawal, Jason Riesa, Dmitry Lepikhin, Richard Tanburn, Srivatsan Srinivasan, Hyeontaek Lim, Sarah Hodkinson, Pranav Shyam, Johan Ferret, Steven Hand, Ankush Garg, Tom Le Paine, Jian Li, Yujia Li, Minh Giang, Alexander Neitz, Zaheer Abbas, Sarah York, Machel Reid, Elizabeth Cole, Aakanksha Chowdhery, Dipanjan Das, Dominika Rogozińska, Vitaly Nikolaev, Pablo Sprechmann, Zachary Nado, Lukas Zilka, Flavien Prost, Luheng He, Marianne Monteiro, Gaurav Mishra, Chris Welty, Josh Newlan, Dawei Jia, Miltiadis Allamanis, Clara Huiyi Hu, Raoul de Liedekerke, Justin Gilmer, Carl Saroufim, Shruti Rijhwani, Shaobo Hou, Disha Shrivastava, Anirudh Baddepudi, Alex Goldin, Adnan Ozturel, Albin Cassirer, Yunhan Xu, Daniel Sohn, Devendra Sachan, Reinald Kim Amplayo, Craig Swanson, Dessie Petrova, Shashi Narayan, Arthur Guez, Siddhartha Brahma, Jessica Landon, Miteyan Patel, Ruizhe Zhao, Kevin Villela, Luyu Wang, Wenhao Jia, Matthew Rahtz, Mai Giménez, Legg Yeung, Hanzhao Lin, James Keeling, Petko Georgiev, Diana Mincu, Boxi Wu, Salem Haykal, Rachel Saputro, Kiran Vodrahalli, James Qin, Zeynep Cankara, Abhanshu Sharma, Nick Fernando, Will Hawkins, Behnam Neyshabur, Solomon Kim, Adrian Hutter, Priyanka Agrawal, Alex Castro-Ros, George van den Driessche, Tao Wang, Fan Yang, Shuo yiin Chang, Paul Komarek, Ross McIlroy, Mario Lučić, Guodong Zhang, Wael Farhan, Michael Sharman, Paul Natsev, Paul Michel, Yong Cheng, Yamini Bansal, Siyuan Qiao, Kris Cao, Siamak Shakeri, Christina Butterfield, Justin Chung, Paul Kishan Rubenstein, Shivani Agrawal, Arthur Mensch, Kedar Soparkar, Karel Lenc, Timothy Chung, Aedan Pope, Loren Maggiore, Jackie Kay, Priya Jhakra, Shibo Wang, Joshua Maynez, Mary Phuong, Taylor Tobin, Andrea Tacchetti, Maja Trebacz, Kevin Robinson, Yash Katariya, Sebastian Riedel, Paige Bailey, Kefan Xiao, Nimesh Ghelani, Lora Aroyo, Ambrose Slone, Neil Houlsby, Xuehan Xiong, Zhen Yang, Elena Gribovskaya, Jonas Adler, Mateo Wirth, Lisa Lee, Music Li, Thais Kagohara, Jay Pavagadhi, Sophie Bridgers, Anna Bortsova, Sanjay Ghemawat, Zafarali Ahmed, Tianqi Liu, Richard Powell, Vijay Bolina, Mariko Iinuma, Polina Zablotskaia, James Besley, Da-Woon Chung, Timothy Dozat, Ramona Comanescu, Xiance Si, Jeremy Greer, Guolong Su, Martin Polacek, Raphaël Lopez Kaufman, Simon Tokumine, Hexiang Hu, Elena Buchatskaya, Yingjie Miao, Mohamed Elhawaty, Aditya Siddhant, Nenad Tomasev, Jinwei Xing, Christina Greer, Helen Miller, Shereen Ashraf, Aurko Roy, Zizhao Zhang, Ada Ma, Angelos Filos, Milos Besta, Rory Blevins, Ted Klimenko, Chih-Kuan Yeh, Soravit Changpinyo, Jiaqi Mu, Oscar Chang, Mantas Pajarskas, Carrie Muir, Vered Cohen, Charline Le Lan, Krishna Haridasan, Amit Marathe, Steven Hansen, Sholto Douglas, Rajkumar Samuel, Mingqiu Wang, Sophia Austin, Chang Lan, Jiepu Jiang, Justin Chiu, Jaime Alonso Lorenzo, Lars Lowe Sjösund, Sébastien Cevey, Zach Gleicher, Thi Avrahami, Anudhyan Boral, Hansa Srinivasan, Vittorio Selo, Rhys May, Konstantinos Aisopos, Léonard Hussenot, Livio Baldini Soares, Kate Baumli, Michael B. Chang, Adrià Recasens, Ben Caine, Alexander Pritzel, Filip Pavetic, Fabio Pardo, Anita Gergely, Justin Frye, Vinay Ramasesh, Dan Horgan, Kartikeya Badola, Nora Kassner, Subhrajit Roy, Ethan Dyer, Víctor Campos, Alex Tomala, Yunhao Tang, Dalia El Badawy, Elspeth White, Basil Mustafa, Oran Lang, Abhishek Jindal, Sharad Vikram, Zhitao Gong, Sergi Caelles, Ross Hemsley, Gregory Thornton, Fangxiaoyu Feng, Wojciech Stokowiec, Ce Zheng, Phoebe Thacker, Çağlar Ünlü, Zhishuai Zhang, Mohammad Saleh, James Svensson, Max Bileschi, Piyush Patil, Ankesh Anand, Roman Ring, Katerina Tsihlas, Arpi Vezer, Marco Selvi, Toby Shevlane, Mikel Rodriguez, Tom Kwiatkowski, Samira Daruki, Keran Rong, Allan Dafoe, Nicholas FitzGerald, Keren Gu-Lemberg, Mina Khan, Lisa Anne Hendricks, Marie Pellat, Vladimir Feinberg, James Cobon-Kerr, Tara Sainath, Maribeth Rauh, Sayed Hadi Hashemi, Richard Ives, Yana Hasson, YaGuang Li, Eric Noland, Yuan Cao, Nathan Byrd, Le Hou, Qingze Wang, Thibault Sottiaux, Michela Paganini, Jean-Baptiste Lespiau, Alexandre Moufarek, Samer Hassan, Kaushik Shivakumar, Joost van Amersfoort, Amol Mandhane, Pratik Joshi, Anirudh Goyal, Matthew Tung, Andrew Brock, Hannah Sheahan, Vedant Misra, Cheng Li, Nemanja Rakićević, Mostafa Dehghani, Fangyu Liu, Sid Mittal, Junhyuk Oh, Seb Noury, Eren Sezener, Fantine Huot, Matthew Lamm, Nicola De Cao, Charlie Chen, Gamaleldin Elsayed, Ed Chi, Mahdis Mahdieh, Ian Tenney, Nan Hua, Ivan Petrychenko, Patrick Kane, Dylan Scandinaro, Rishub Jain, Jonathan Uesato, Romina Datta, Adam Sadovsky, Oskar Bunyan, Dominik Rabiej, Shimu Wu, John Zhang, Gautam Vasudevan, Edouard Leurent, Mahmoud Alnahlawi, Ionut Georgescu, Nan Wei, Ivy Zheng, Betty Chan, Pam G Rabinovitch, Piotr Stanczyk, Ye Zhang, David Steiner, Subhajit Naskar, Michael Azzam, Matthew Johnson, Adam Paszke, Chung-Cheng Chiu, Jaume Sanchez Elias, Afroz Mohiuddin, Faizan Muhammad, Jin Miao, Andrew Lee, Nino Vieillard, Sahitya Potluri, Jane Park, Elnaz Davoodi, Jiageng Zhang, Jeff Stanway, Drew Garmon, Abhijit Karmarkar, Zhe Dong, Jong Lee, Aviral Kumar, Luowei Zhou, Jonathan Evens, William Isaac, Zhe Chen, Johnson Jia, Anselm Levskaya, Zhenkai Zhu, Chris Gorgolewski, Peter Grabowski, Yu Mao, Alberto Magni, Kaisheng Yao, Javier Snaider, Norman Casagrande, Paul Suganthan, Evan Palmer, Geoffrey Irving, Edward Loper, Manaal Faruqui, Isha Arkatkar, Nanxin Chen, Izhak Shafran, Michael Fink, Alfonso Castaño, Irene Giannoumis, Wooyeol Kim, Mikołaj Rybiński, Ashwin Sreevatsa, Jennifer Prendki, David Soergel, Adrian Goedeckemeyer, Willi Gierke, Mohsen Jafari, Meenu Gaba, Jeremy Wiesner, Diana Gage Wright, Yawen Wei, Harsha Vashisht, Yana Kulizhskaya, Jay Hoover, Maigo Le, Lu Li, Chimezie Iwuanyanwu, Lu Liu, Kevin Ramirez, Andrey Khorlin, Albert Cui, Tian LIN, Marin Georgiev, Marcus Wu, Ricardo Aguilar, Keith Pallo, Abhishek Chakladar, Alena Repina, Xihui Wu, Tom van der Weide, Priya Ponnapalli, Caroline Kaplan, Jiri Simsa, Shuangfeng Li, Olivier Dousse, Fan Yang, Jeff Piper, Nathan Ie, Minnie Lui, Rama Pasumarthi, Nathan Lintz, Anitha Vijayakumar, Lam Nguyen Thiet, Daniel Andor, Pedro Valenzuela, Cosmin Paduraru, Daiyi Peng, Katherine Lee, Shuyuan Zhang, Somer Greene, Duc Dung Nguyen, Paula Kurylowicz, Sarmishta Velury, Sebastian Krause, Cassidy Hardin, Lucas Dixon, Lili Janzer, Kiam Choo, Ziqiang Feng, Biao Zhang, Achintya Singhal, Tejasi Latkar, Mingyang Zhang, Quoc Le, Elena Allica Abellan, Dayou Du, Dan McKinnon, Natasha Antropova, Tolga Bolukbasi, Orgad Keller, David Reid, Daniel Finchelstein, Maria Abi Raad, Remi Crocker, Peter Hawkins, Robert Dadashi, Colin Gaffney, Sid Lall, Ken Franko, Egor Filonov, Anna Bulanova, Rémi Leblond, Vikas Yadav, Shirley Chung, Harry Askham, Luis C. Cobo, Kelvin Xu, Felix Fischer, Jun Xu, Christina Sorokin, Chris Alberti, Chu-Cheng Lin, Colin Evans, Hao Zhou, Alek Dimitriev, Hannah Forbes, Dylan Banarse, Zora Tung, Jeremiah Liu, Mark Omernick, Colton Bishop, Chintu Kumar, Rachel Sterneck, Ryan Foley, Rohan Jain, Swaroop Mishra, Jiawei Xia, Taylor Bos, Geoffrey Cideron, Ehsan Amid, Francesco Piccinno, Xingyu Wang, Praseem Banzal, Petru Gurita, Hila Noga, Premal Shah, Daniel J. Mankowitz, Alex Polozov, Nate Kushman, Victoria Krakovna, Sasha Brown, MohammadHossein Bateni, Dennis Duan, Vlad Firoiu, Meghana Thotakuri, Tom Natan, Anhad Mohananey, Matthieu Geist, Sidharth Mudgal, Sertan Girgin, Hui Li, Jiayu Ye, Ofir Roval, Reiko Tojo, Michael Kwong, James Lee-Thorp, Christopher Yew, Quan Yuan, Sumit Bagri, Danila Sinopalnikov, Sabela Ramos, John Mellor, Abhishek Sharma, Aliaksei Severyn, Jonathan Lai, Kathy Wu, Heng-Tze Cheng, David Miller, Nicolas Sonnerat, Denis Vnukov, Rory Greig, Jennifer Beattie, Emily Caveness, Libin Bai, Julian Eisenschlos, Alex Korchemniy, Tomy Tsai, Mimi Jasarevic, Weize Kong, Phuong Dao, Zeyu Zheng, Frederick Liu, Fan Yang, Rui Zhu, Mark Geller, Tian Huey Teh, Jason Sanmiya, Evgeny Gladchenko, Nejc Trdin, Andrei Sozanschi, Daniel Toyama, Evan Rosen, Sasan Tavakkol, Linting Xue, Chen Elkind, Oliver Woodman, John Carpenter, George Papamakarios, Rupert Kemp, Sushant Kafle, Tanya Grunina, Rishika Sinha, Alice Talbert, Abhimanyu Goyal, Diane Wu, Denese Owusu-Afriyie, Cosmo Du, Chloe Thornton, Jordi Pont-Tuset, Pradyumna Narayana, Jing Li, Sabaer Fatehi, John Wieting, Omar Ajmeri, Benigno Uria, Tao Zhu, Yeongil Ko, Laura Knight, Amélie Héliou, Ning Niu, Shane Gu, Chenxi Pang, Dustin Tran, Yeqing Li, Nir Levine, Ariel Stolovich, Norbert Kalb, Rebeca Santamaria-Fernandez, Sonam Goenka, Wenny Yustalim, Robin Strudel, Ali Elqursh, Balaji Lakshminarayanan, Charlie Deck, Shyam Upadhyay, Hyo Lee, Mike Dusenberry, Zonglin Li, Xuezhi Wang, Kyle Levin, Raphael Hoffmann, Dan Holtmann-Rice, Olivier Bachem, Summer Yue, Sho Arora, Eric Malmi, Daniil Mirylenka, Qijun Tan, Christy Koh, Soheil Hassas Yeganeh, Siim Põder, Steven Zheng, Francesco Pongetti, Mukarram Tariq, Yanhua Sun, Lucian Ionita, Mojtaba Seyedhosseini, Pouya Tafti, Ragha Kotikalapudi, Zhiyu Liu, Anmol Gulati, Jasmine Liu, Xinyu Ye, Bart Chrzaszcz, Lily Wang, Nikhil Sethi, Tianrun Li, Ben Brown, Shreya Singh, Wei Fan, Aaron Parisi, Joe Stanton, Chenkai Kuang, Vinod Koverkathu, Christopher A. Choquette-Choo, Yunjie Li, TJ Lu, Abe Ittycheriah, Prakash Shroff, Pei Sun, Mani Varadarajan, Sanaz Bahargam, Rob Willoughby, David Gaddy, Ishita Dasgupta, Guillaume Desjardins, Marco Cornero, Brona Robenek, Bhavishya Mittal, Ben Albrecht, Ashish Shenoy, Fedor Moiseev, Henrik Jacobsson, Alireza Ghaffarkhah, Morgane Rivière, Alanna Walton, Clément Crepy, Alicia Parrish, Yuan Liu, Zongwei Zhou, Clement Farabet, Carey Radebaugh, Praveen Srinivasan, Claudia van der Salm, Andreas Fidjeland, Salvatore Scellato, Eri Latorre-Chimoto, Hanna Klimczak-Plucińska, David Bridson, Dario de Cesare, Tom Hudson, Piermaria Mendolicchio, Lexi Walker, Alex Morris, Ivo Penchev, Matthew Mauger, Alexey Guseynov, Alison Reid, Seth Odoom, Lucia Loher, Victor Cotruta, Madhavi Yenugula, Dominik Grewe, Anastasia Petrushkina, Tom Duerig, Antonio Sanchez, Steve Yadlowsky, Amy Shen, Amir Globerson, Adam Kurzrok, Lynette Webb, Sahil Dua, Dong Li, Preethi Lahoti, Surya Bhupatiraju, Dan Hurt, Haroon Qureshi, Ananth Agarwal, Tomer Shani, Matan Eyal, Anuj Khare, Shreyas Rammohan Belle, Lei Wang, Chetan Tekur, Mihir Sanjay Kale, Jinliang Wei, Ruoxin Sang, Brennan Saeta, Tyler Liechty, Yi Sun,

Yao Zhao, Stephan Lee, Pandu Nayak, Doug Fritz, Manish Reddy Vuyyuru, John Aslanides, Nidhi Vyas, Martin Wicke, Xiao Ma, Taylan Bilal, Evgenii Eltyshev, Daniel Balle, Nina Martin, Hardie Cate, James Manyika, Keyvan Amiri, Yelin Kim, Xi Xiong, Kai Kang, Florian Luisier, Nilesh Tripuraneni, David Madras, Mandy Guo, Austin Waters, Oliver Wang, Joshua Ainslie, Jason Baldridge, Han Zhang, Garima Pruthi, Jakob Bauer, Feng Yang, Riham Mansour, Jason Gelman, Yang Xu, George Polovets, Ji Liu, Honglong Cai, Warren Chen, XiangHai Sheng, Emily Xue, Sherjil Ozair, Adams Yu, Christof Angermueller, Xiaowei Li, Weiren Wang, Julia Wiesinger, Emmanouil Koukoumidis, Yuan Tian, Anand Iyer, Madhu Gurumurthy, Mark Goldenson, Parashar Shah, MK Blake, Hongkun Yu, Anthony Urbanowicz, Jennimaria Palomaki, Chrisantha Fernando, Kevin Brooks, Ken Durden, Harsh Mehta, Nikola Momchev, Elahe Rahimtoroghi, Maria Georgaki, Amit Raul, Sebastian Ruder, Morgan Redshaw, Jinhyuk Lee, Komal Jalan, Dinghua Li, Ginger Perng, Blake Hechtman, Parker Schuh, Milad Nasr, Mia Chen, Kieran Milan, Vladimir Mikulik, Trevor Strohman, Juliana Franco, Tim Green, Demis Hassabis, Koray Kavukcuoglu, Jeffrey Dean, and Oriol Vinyals. 2023. Gemini: A family of highly capable multimodal models.

Jordan T. Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. 2020. Deep Batch Active Learning by Diverse, Uncertain Gradient Lower Bounds. In *International Conference on Learning Representations*, pages 1–26, Online.

Esmat Babaii and Hasan Ansary. 2001. The C-test: a valid operationalization of reduced redundancy principle? *System*, 29(2):209–219.

Ryan S Baker. 2016. Stupid tutoring systems, intelligent humans. *International Journal of Artificial Intelligence in Education*, 26(2):600–614.

Alexis Baladón, Ignacio Sastre, Luis Chiruzzo, and Aiala Rosá. 2023. RETUYT-InCo at BEA 2023 shared task: Tuning open-source LLMs for generating teacher responses. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, pages 756–765, Toronto, Canada. Association for Computational Linguistics.

Juliana V Baldo, Nina F Dronkers, David Wilkins, Carl Ludy, Patricia Raskin, and Jiye Kim. 2005. Is problem solving dependent on language? *Brain and language*, 92(3):240–250.

Robert Baldock, Hartmut Maennel, and Behnam Neyshabur. 2021. Deep Learning Through the Lens of Example Difficulty. In *Advances in Neural Information Processing Systems*, volume 34, pages 10876–10889. Curran Associates, Inc.

Marta Bañón, Pinzhen Chen, Barry Haddow, Kenneth Heafield, Hieu Hoang, Miquel Esplà-Gomis, Mikel L. Forcada, Amir Kamran, Faheem Kirefu, Philipp Koehn, Sergio Ortiz Rojas, Leopoldo Pla Sempere, Gema Ramírez-Sánchez, Elsa Sarrías, Marek Strelec, Brian Thompson, William Waites, Dion Wiggins, and Jaume Zaragoza. 2020. ParaCrawl: Web-scale acquisition of parallel corpora. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4555–4567, Online. Association for Computational Linguistics.

Connor Baumler, Anna Sotnikova, and Hal Daumé III. 2023. Which examples should be multiply annotated? active learning when annotators may disagree. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10352–10371, Toronto, Canada. Association for Computational Linguistics.

Petra Saskia Bayerl and Karsten Ingmar Paul. 2011. What Determines Inter-Coder Agreement in Manual Annotations? A Meta-Analytic Investigation. *Computational Linguistics*, 37(4):699–725.

Lisa Beinborn, Torsten Zesch, and Iryna Gurevych. 2014a. Predicting the Difficulty of Language Proficiency Tests. *Transactions of the Association for Computational Linguistics*, 2:517–529.

Lisa Beinborn, Torsten Zesch, and Iryna Gurevych. 2014b. Readability for foreign language learning: The importance of cognates. *ITL-International Journal of Applied Linguistics*, 165(2):136–162.

Lisa Beinborn, Torsten Zesch, and Iryna Gurevych. 2015. Candidate evaluation strategies for improved difficulty prediction of language tests. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 1–11, Denver, Colorado. Association for Computational Linguistics.

Lisa Marina Beinborn. 2016. *Predicting and manipulating the difficulty of text-completion exercises for language learning*. Ph.D. thesis, Technische Universität Darmstadt.

Luca Benedetto, Giovanni Aradelli, Paolo Cremonesi, Andrea Cappelli, Andrea Giussani, and Roberto Turrin. 2021. On the application of transformers for estimating the difficulty of multiple-choice questions from text. In *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 147–157, Online. Association for Computational Linguistics.

Luca Benedetto, Paolo Cremonesi, Andrew Caines, Paula Buttery, Andrea Cappelli, Andrea Giussani, and Roberto Turrin. 2023. A survey on recent approaches to question difficulty estimation from text. *ACM Comput. Surv.*, 55(9).

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference On Machine Learning*, pages 41–48, Montreal, Canada.

Elke Cases Berbel. 2020. Challenges and difficulties of translation and interpreting in the migration and refugee crisis in germany. *Open Linguistics*, 6(1):162–170.

Allan Birnbaum. 1968. Some latent trait models and their use in inferring an examinee's ability. *Statistical theories of mental test scores.*

Semere Kiros Bitew, Johannes Deleu, A. Seza Doğruöz, Chris Develder, and Thomas Demeester. 2023. Learning from partially annotated data: Example-aware creation of gap-filling exercises for language learning. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, pages 598–609, Toronto, Canada. Association for Computational Linguistics.

Ana Marín Blanco, Göran Bostedt, Dirk Michel-Schertges, and Sabrina Wüllner. 2023. Studying teacher shortages: Theoretical perspectives and methodological approaches. *Journal of Pedagogical Research*, 7(1).

Zalán Bodó, Zsolt Minier, and Lehel Csató. 2011. Active Learning with Clustering. In *Active Learning and Experimental Design workshop In conjunction with AISTATS 2010*, pages 127–139. JMLR Workshop and Conference Proceedings.

C.C. Bonwell, J.A. Eison, Association for the Study of Higher Education, ERIC Clearinghouse on Higher Education, DC. School of Education George Washington Univ., Washington, and Human Development. 1991. *Active Learning: Creating Excitement in the Classroom. 1991 ASHE-ERIC Higher Education Reports*. ERIC Clearinghouse on Higher Education.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal.

Inês Carvalho, Ana Ramires, and Montserrat Iglesias. 2023. Attitudes towards machine translation and languages among travelers. *Information Technology & Tourism*, pages 1–30.

Jon Chamberlain, Karën Fort, Udo Kruschwitz, Mathieu Lafourcade, and Massimo Poesio. 2013. Using games to create language resources: Successes and limitations of the approach. *The People's Web Meets NLP: Collaboratively Constructed Language Resources*, pages 3–44.

Branden Chan, Stefan Schweter, and Timo Möller. 2020. German's next language model. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6788–6796, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Sophia Chan, Swapna Somasundaran, Debanjan Ghosh, and Mengxuan Zhao. 2022. AGReE: A system for generating automated grammar reading exercises. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–177, Abu Dhabi, UAE. Association for Computational Linguistics.

C. A. Chapelle. 1994. Are C-tests valid measures for L2 vocabulary research? *Second Language Research*, 10(2):157–187.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIG-KDD international conference on knowledge discovery and data mining*, pages 785–794.

Min Chi, Kurt VanLehn, and Diane Litman. 2010. Do micro-level tutorial decisions matter: Applying reinforcement learning to induce pedagogical tutorial tactics. In *Intelligent Tutoring Systems*, pages 224–234, Berlin, Heidelberg. Springer Berlin Heidelberg.

Barry R. Chiswick. 1991. Speaking, reading, and earnings among low-skilled immigrants. *Journal of Labor Economics*, 9(2):149–170.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.

Christopher Cleary. 1988. The c-test in english: left-hand deletions. *Regional Language Centre (RELC)*, 19(2):26–35.

Virginia Clinton-Lisell. 2021. Open pedagogy: A systematic review of empirical findings. *Journal of Learning for Development*, 8(2):255–268.

Vivian J Cook. 1992. Evidence for multicompetence. *Language learning*, 42(4):557–591.

Albert T Corbett and John R Anderson. 1994. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4(4):253–278.

Corinna Cortes, Mehryar Mohri, Michael Riley, and Afshin Rostamizadeh. 2008. Sample Selection Bias Correction Theory. In *Algorithmic Learning Theory*, pages 38–53, Berlin, Heidelberg. Springer Berlin Heidelberg.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20:273–297.

Council of Europe. 2001. *Common European framework of reference for languages: Learning, teaching, assessment*. Cambridge University Press.

Council of Europe. 2002. Presidency Conclusions. Barcelona European Council 15 and 16 March 2002. Report SN 100/1/02 REV 1, Council of the European Union.

A. Garr Cranney. 1972. The construction of two types of cloze reading tests for college students. *Journal of Reading Behavior*, 5(1):60–64.

Kevin Crowston. 2012. Amazon mechanical turk: A research tool for organizations and information systems scholars. In *Shaping the future of ict research. methods and approaches*, pages 210–221. Springer.

Peng Cui and Mrinmaya Sachan. 2023. Adaptive and personalized exercise generation for online language learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10184–10198, Toronto, Canada. Association for Computational Linguistics.

Jet Cullen and Alan Bryman. 1988. The knowledge acquisition bottleneck: time for reassessment? *Expert Systems*, 5(3):216–225.

Lennart Delander, Mats Hammarstedt, Jonas MÅnsson, and Erik Nyberg. 2005. Integration of immigrants: The role of language proficiency and experience. *Evaluation Review*, 29(1):24–41. PMID: 15604118.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short*

Papers), pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping. *arXiv preprint arXiv:2002.06305v1*.

George Duenas, Sergio Jimenez, and Geral Mateus Ferro. 2023. You've got a friend in ... a language model? a comparison of explanations of multiple-choice items of reading comprehension between ChatGPT and humans. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, pages 372–381, Toronto, Canada. Association for Computational Linguistics.

Jacquelynne S Eccles. 2005. Subjective task value and the Eccles et al. model of achievement-related choices. *Handbook of Competence and Motivation*, pages 105–121.

Stephanie Eckman, Barbara Plank, and Frauke Kreuter. 2024. The science of data collection: Insights from surveys can improve machine learning models.

Liat Ein-Dor, Alon Halfon, Ariel Gera, Eyal Shnarch, Lena Dankin, Leshem Choshen, Marina Danilevsky, Ranit Aharonov, Yoav Katz, and Noam Slonim. 2020. Active Learning for BERT: An Empirical Study. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7949–7962, Online. Association for Computational Linguistics.

Jeffrey L. Elman. 1993. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99.

Saadia Gutta Essa, Turgay Celik, and Nadia Emelia Human-Hendricks. 2023. Personalized adaptive learning technologies based on machine learning techniques to identify learning styles: A systematic literature review. *IEEE Access*, 11:48392–48409.

Kawin Ethayarajh, Yejin Choi, and Swabha Swayamdipta. 2022. Understanding dataset difficulty with $\mathcal{V}$-usable information. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 5988–6008. PMLR.

EU. 2016. Consolidated text: Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance).

Jerry Alan Fails and Dan R. Olsen. 2003. Interactive machine learning. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*, IUI '03, page 39–45, New York, NY, USA. Association for Computing Machinery.

Hossein Farhady and F Jamali. 2006. Varieties of c-test as measures of general language proficiency. *Twenty-five years of living with applied linguistics: collection of articles*, pages 287–302.

Mariano Felice and Paula Buttery. 2019. Entropy as a Proxy for Gap Complexity in Open Cloze Tests. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 323–327, Varna, Bulgaria.

Mariano Felice, Shiva Taslimipoor, and Paula Buttery. 2022. Constructing open cloze tests using generation and discrimination capabilities of transformers. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1263–1273, Dublin, Ireland. Association for Computational Linguistics.

Fundamental Rights Agency. 2023. Asylum and migration: Progress achieved and remaining challenges. *European Union Agency for Fundamental Rights*.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. 2017. Deep Bayesian Active Learning with Image Data. In *International Conference on Machine Learning*, pages 1183–1192, Sydney, Australia. PMLR.

Isabel O. Gallegos, Ryan A. Rossi, Joe Barrow, Md Mehrab Tanjim, Sungchul Kim, Franck Dernoncourt, Tong Yu, Ruiyi Zhang, and Nesreen K. Ahmed. 2023. Bias and fairness in large language models: A survey.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

Lingyu Gao, Kevin Gimpel, and Arnar Jensson. 2020. Distractor analysis and selection for multiple-choice cloze questions for second-language learners. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 102–114, Seattle, WA, USA, Online. Association for Computational Linguistics.

Emma García and Elaine Weiss. 2019. The teacher shortage is real, large and growing, and worse than we thought. the first report in" the perfect storm in the teacher labor market" series. *Economic policy institute*.

Michele J Gelfand, Sergey Gavrilets, and Nathan Nunn. 2024. Norm dynamics: interdisciplinary perspectives on social norm emergence, persistence, and change. *Annual Review of Psychology*, 75:341–378.

Alex Graves, Marc G. Bellemare, Jacob Menick, Rémi Munos, and Koray Kavukcuoglu. 2017. Automated curriculum learning for neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1311–1320. PMLR.

Rüdiger Grotjahn. 2006. *Der C-Test: Theorie, Empirie, Anwendungen / The C-Test: Theory, Empirical Research, Applications*. Peter Lang Verlag, Berlin, Germany.

Anshita Gupta, Debanjan Mondal, Akshay Krishna Sheshadri, Wenlong Zhao, Xiang Lorraine Li, Sarah Wiegreffe, and Niket Tandon. 2023. Editing common sense in transformers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, page (to appear). Association for Computational Linguistics.

Ronald K Hambleton and Russell W Jones. 1993. Comparison of classical test theory and item response theory and their applications to test development. *Educational Measurement: Issues and Practice*, 12:38–47.

Ronald K Hambleton, Hariharan Swaminathan, and H Jane Rogers. 1991. *Fundamentals of item response theory*, volume 2. Sage.

Christian Haring, Rene Lehmann, Andrea Horbach, and Torsten Zesch. 2021. C-test collector: A proficiency testing application to collect training data for C-tests. In *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 180–184, Online. Association for Computational Linguistics.

Robert Hart. 1981. Language study and the PLATO system. *Studies in language learning*, 3(1):1–24.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. DeBERTa: Decoding-enhanced BERT with Disentangled Attention. In *International Conference on Learning Representations*, pages 1–21.

Tanja Heck and Detmar Meurers. 2022. Parametrizable exercise generation from authentic texts: Effectively targeting the language means on the curriculum. In *Proceedings of the 17th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2022)*, pages 154–166, Seattle, Washington. Association for Computational Linguistics.

Tanja Heck and Detmar Meurers. 2023. Using learning analytics for adaptive exercise generation. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, pages 44–56, Toronto, Canada. Association for Computational Linguistics.

Jennifer Hill and Rahul Simha. 2016. Automatic generation of context-based fill-in-the-blank exercises using co-occurrence likelihoods and google n-grams. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications (BEA)*, pages 23–30, San Diego, CA, USA.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack William Rae, and Laurent Sifre. 2022. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems*, pages 1–15.

Ayako Hoshino, Hiroshi Nakagawa, et al. 2010. Predicting the difficulty of multiple-choice close questions for computer-adaptive testing. *Natural Language Processing and its Applications*, 46:279–292.

Fu-Yuan Hsu, Hahn-Ming Lee, Tao-Hsing Chang, and Yao-Ting Sung. 2018. Automated estimation of item difficulty for multiple-choice tests: An application of word embedding techniques. *Information Processing & Management*, 54(6):969–984.

Nathan Hu, Eric Mitchell, Christopher Manning, and Chelsea Finn. 2023. Meta-learning online adaptation of language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4418–4432, Singapore. Association for Computational Linguistics.

Pia Hummelsberger, Timo K Koch, Sabrina Rauh, Julia Dorn, Eva Lermer, Martina Raue, Matthias F C Hudecek, Andreas Schicho, Errol Colak, Marzyeh Ghassemi, and Susanne Gaube. 2023. Insights on the current state and future outlook of ai in health care: Expert interview study. *JMIR AI*, 2:e47353.

Jan Hylén. 2020. Open educational resources: Opportunities and challenges. *Centre for Educational Research and Innovation*.

Knud Illeris. 2003. Towards a contemporary and comprehensive theory of learning. *International journal of lifelong education*, 22(4):396–406.

Joseph Marvin Imperial and Harish Tayyar Madabushi. 2023. Flesch or fumble? evaluating readability standard alignment of instruction-tuned language models. *arXiv preprint arXiv:2309.05454*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12).

E.A. Jongsma. 1971. *The Cloze Procedure as a Teaching Technique by Eugene Jongsma*. Where do we go? International Reading Association.

Daniel Jurafsky and James H. Martin. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 1st edition. Prentice Hall PTR, USA.

Tadamitsu Kamimoto. 1993. Tailoring the test to fit the students : Improvement of the c-test through classical item analysis. *Language Laboratory*, 30:47–61.

Siddharth Karamcheti, Ranjay Krishna, Li Fei-Fei, and Christopher Manning. 2021. Mind your outliers! investigating the negative impact of outliers on active learning for visual question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7265–7281, Online.

Anisia Katinskaia and Roman Yangarber. 2023. Grammatical error correction for sentence-level assessment in language learning. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, pages 488–502, Toronto, Canada. Association for Computational Linguistics.

Parneet Kaur, Harish Kumar, and Sakshi Kaushal. 2023. Technology-assisted language learning adaptive systems: A comprehensive review. *International Journal of Cognitive Computing in Engineering*, 4:301–313.

A. V. Kelly. 1977. *The Curriculum : Theory and Practice*. SAGE Publications, Thousand Oaks, CA.

M.M. Kenning and M.J. Kenning. 1990. *Computers and Language Learning: Current Theory and Practice*. Computers and their applications. Ellis Horwood.

Osman Doruk Kicikoglu, Richard Bartle, Jon Chamberlain, Silviu Paun, and Massimo Poesio. 2020. Aggregation Driven Progression System for GWAPs. In *Workshop on Games and Natural Language Processing*, pages 79–84, Marseille, France.

Andreas Kirsch, Tom Rainforth, and Yarin Gal. 2021. Test distribution-aware active learning: A principled approach against distribution shift and outliers. *arXiv preprint arXiv:2106.11719*.

Andreas Kirsch, Joost van Amersfoort, and Yarin Gal. 2019. BatchBALD: Efficient and Diverse Batch Acquisition for Deep Bayesian Active Learning. In *Advances in Neural Information Processing Systems*, pages 7026–7037, Vancouver, Canada.

Wolfgang Klein. 1986. *Second language acquisition*. Cambridge University Press.

Christine Klein-Braley and Ulrich Raatz. 1982. Der C-Test: ein neuer Ansatz zur Messung allgemeiner Sprachbeherrschung. *AKS-Rundbrief*, 4:23–37.

Jan-Christoph Klie, Michael Bugert, Beto Boullosa, Richard Eckart de Castilho, and Iryna Gurevych. 2018. The INCEpTION Platform: Machine-Assisted and Knowledge-Oriented Interactive Annotation. In *Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations*, pages 5–9. Association for Computational Linguistics.

Jan-Christoph Klie, Richard Eckart de Castilho, and Iryna Gurevych. 2024. Analyzing Dataset Annotation Quality Management in the Wild. *Computational Linguistics*, pages 1–48.

Jan-Christoph Klie, Richard Eckart de Castilho, and Iryna Gurevych. 2020. From Zero to Hero: Human-In-The-Loop Entity Linking in Low Resource Domains. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6982–6993, Online. Association for Computational Linguistics.

Tom Kocmi, Eleftherios Avramidis, Rachel Bawden, Ondřej Bojar, Anton Dvorkovich, Christian Federmann, Mark Fishel, Markus Freitag, Thamme Gowda, Roman Grundkiewicz, Barry Haddow, Philipp Koehn, Benjamin Marie, Christof Monz, Makoto Morishita, Kenton Murray, Makoto Nagata, Toshiaki Nakazawa, Martin Popel, Maja Popović, and Mariya Shmatova. 2023. Findings of the 2023 conference on machine translation (WMT23): LLMs are here but not quite there yet. In *Proceedings of the Eighth Conference on Machine Translation*, pages 1–42, Singapore. Association for Computational Linguistics.

Tom Kocmi and Ondřej Bojar. 2017. Curriculum learning and minibatch bucketing in neural machine translation. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 379–386, Varna, Bulgaria. INCOMA Ltd.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. 2020. Concept bottleneck models. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5338–5348. PMLR.

Lucas Kohnke, Benjamin Luke Moorhouse, and Di Zou. 2023. Chatgpt for language teaching and learning. *Regional Language Centre (RELC)*, 54(2):537–550.

Daniel Kottke, Adrian Calma, Denis Huseljic, Georg Krempl, and Bernhard Sick. 2017. Challenges of reliable, realistic and comparable active learning evaluation. In *Proceedings of the Workshop and Tutorial on Interactive Adaptive Learning co-located with European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2017), Skopje, Macedonia, September 18, 2017.*, pages 2–14.

Stephen Krashen. 1982. *Principles and Practice in Second Language Acquisition*. Pergamon Press, Oxford New York.

Julia Kreutzer, Isaac Caswell, Lisa Wang, Ahsan Wahab, Daan van Esch, Nasanbayar Ulzii-Orshikh, Allahsera Tapo, Nishant Subramani, Artem Sokolov, Claytone Sikasote, Monang Setyawan, Supheakmungkol Sarin, Sokhar Samb, Benoît Sagot, Clara Rivera, Annette Rios, Isabel Papadimitriou, Salomey Osei, Pedro Ortiz Suarez, Iroro Orife, Kelechi Ogueji, Andre Niyongabo Rubungo, Toan Q. Nguyen, Mathias Müller, André Müller, Shamsuddeen Hassan Muhammad, Nanda Muhammad, Ayanda Mnyakeni, Jamshidbek Mirzakhalov, Tapiwanashe Matangira, Colin Leong, Nze Lawson, Sneha Kudugunta, Yacine Jernite, Mathias Jenny, Orhan Firat, Bonaventure F. P. Dossou, Sakhile Dlamini, Nisansa de Silva, Sakine Çabuk Ballı, Stella Biderman, Alessia Battisti, Ahmed Baruwa, Ankur Bapna, Pallavi Baljekar, Israel Abebe Azime, Ayodele Awokoya, Duygu Ataman, Orevaoghene Ahia, Oghenefego Ahia, Sweta Agrawal, and Mofetoluwa Adeyemi. 2022. Quality at a Glance: An Audit of Web-Crawled Multilingual Datasets. *Transactions of the Association for Computational Linguistics*, 10:50–72.

M. Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, volume 23, pages 1189–1197.

Jonathan K. Kummerfeld. 2021. Quantifying and avoiding unfair qualification labour in crowdsourcing. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 343–349, Online. Association for Computational Linguistics.

Batia Laufer and Paul Nation. 1999. A vocabulary-size test of controlled productive ability. *Language Testing*, 16(1):33–51.

Antonio Laverghetta Jr. and John Licato. 2023. Generating better items for cognitive assessments using large language models. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, pages 414–428, Toronto, Canada. Association for Computational Linguistics.

Ji-Ung Lee, Jan-Christoph Klie, and Iryna Gurevych. 2022a. Annotation Curricula to Implicitly Train Non-Expert Annotators. *Computational Linguistics*, 48(2):343–373.

Ji-Ung Lee, Christian M. Meyer, and Iryna Gurevych. 2020. Empowering Active Learning to Jointly Optimize System and User Demands. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4233–4247, Online. Association for Computational Linguistics.

Ji-Ung Lee, Haritz Puerto, Betty van Aken, Yuki Arase, Jessica Zosa Forde, Leon Derczynski, Andreas Rücklé, Iryna Gurevych, Roy Schwartz, Emma Strubell, and Jesse Dodge. 2023. Surveying (Dis)Parities and Concerns of Compute Hungry NLP Research. *arXiv preprint arXiv:2306.16900*.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2022b. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8424–8445, Dublin, Ireland. Association for Computational Linguistics.

Michael Levy. 1997. *Computer-assisted language learning: Context and conceptualization*. Oxford University Press.

David D. Lewis and William A. Gale. 1994. A Sequential Algorithm for Training Text Classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 3–12, Dublin, Ireland.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Belinda Z. Li, Gabriel Stanovsky, and Luke Zettlemoyer. 2020. Active learning for coreference resolution using discrete annotation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8320–8331, Online.

Chen Liang, Xiao Yang, Neisarg Dave, Drew Wham, Bart Pursel, and C. Lee Giles. 2018. Distractor generation for multiple choice questions using learning to rank. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 284–290, New Orleans, Louisiana. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Christopher Lin, M Mausam, and Daniel Weld. 2016. Re-active learning: Active learning with relabeling. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1):1845–1852.

Wim J. Linden and Ronald K. Hambleton. 1997. *Handbook of Modern Item Response Theory*. Springer, New York, NY.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In *Advances in Neural Information Processing Systems*, volume 35, pages 1950–1965. Curran Associates, Inc.

Ming Liu, Wray Buntine, and Gholamreza Haffari. 2018. Learning to Actively Learn Neural Machine Translation. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 334–344, Brussels, Belgium. Association for Computational Linguistics.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 55(9).

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Mengsay Loem, Masahiro Kaneko, Sho Takase, and Naoaki Okazaki. 2023. Exploring effectiveness of GPT-3 in grammatical error correction: A study on performance and controllability in prompt-based methods. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, pages 205–219, Toronto, Canada. Association for Computational Linguistics.

Frederic M Lord and Melvin R Novick. 1968. *Statistical theories of mental test scores*. Addison-Wesley.

David Lowell, Zachary C. Lipton, and Byron C. Wallace. 2019. Practical Obstacles to Deploying Active Learning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 21–30, Hong Kong, China.

Verena Lyding, Lionel Nicolas, and Alexander König. 2022. About the applicability of combining implicit crowdsourcing and language learning for the collection of NLP datasets. In *Proceedings of the 2nd Workshop on Novel Incentives in Data Collection from People: models, implementations, challenges and results within LREC 2022*, pages 46–57, Marseille, France. European Language Resources Association.

Chris Madge, Juntao Yu, Jon Chamberlain, Udo Kruschwitz, Silviu Paun, and Massimo Poesio. 2019. Progression in a Language Annotation Game with a Purpose. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, pages 77–85, Honolulu, Hawaii, USA.

Katerina Margatina and Nikolaos Aletras. 2023. On the limitations of simulating active learning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4402–4419, Toronto, Canada. Association for Computational Linguistics.

Katerina Margatina, Loic Barrault, and Nikolaos Aletras. 2022. On the importance of effectively adapting pretrained language models for active learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 825–836, Dublin, Ireland. Association for Computational Linguistics.

Katerina Margatina, Giorgos Vernikos, Loïc Barrault, and Nikolaos Aletras. 2021. Active Learning by Acquiring Contrastive Examples. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 650–663, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Fernand Marty. 1981. Reflections on the use of computers in second-language acqusition — I. *System*, 9(2):85–98.

Arya D. McCarthy, Kevin P. Yancey, Geoffrey T. LaFlair, Jesse Egbert, Manqian Liao, and Burr Settles. 2021. Jump-starting item parameters for adaptive language tests. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 883–899, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Todd McKay. 2019. *More on the validity and reliability of C-test scores: a meta-analysis of C-test studies.* Ph.D. thesis, Georgetown University.

Walter McManus, William Gould, and Finis Welch. 1983. Earnings of hispanic men: The role of english language proficiency. *Journal of Labor Economics*, 1(2):101–130.

Alan Mead and Adam W. Meade. 2010. Ctt and irt 1 test construction using ctt and irt with unrepresentative samples. In *Annual meeting of the Society for Industrial and Organizational Psychology*, volume 56, Atlanta, GA.

Niklas Meyer, Michael Wojatzki, and Torsten Zesch. 2016. Validating bundled gap filling – empirical evidence for ambiguity reduction and language proficiency testing capabilities. In *Proceedings of the joint workshop on NLP for Computer Assisted Language Learning and NLP for Language Acquisition*, pages 51–59, Umeå, Sweden. LiU Electronic Press.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Swaroop Mishra and Bhavdeep Singh Sachdeva. 2020. Do we need to create big datasets to learn a task? In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, pages 169–173, Online. Association for Computational Linguistics.

Ruslan Mitkov and Le An Ha. 2003. Computer-aided generation of multiple-choice tests. In *Proceedings of the HLT-NAACL 03 Workshop on Building Educational Applications Using Natural Language Processing*, pages 17–22.

Eduardo Mosqueira-Rey, Elena Hernández-Pereira, David Alonso-Ríos, José Bobes-Bascarán, and Ángel Fernández-Leal. 2023. Human-in-the-loop machine learning: a state of the art. *Artificial Intelligence Review*, 56(4):3005–3054.

Marvin Müller, Nicholas Frick, and Joachim Metternich. 2021. Wissen aus betrieblichen chats nachhaltig nutzen. *wt Werkstattstechnik online*, 111(1-2):93–96.

Tom Murray. 1999. Authoring Intelligent Tutoring Systems: An analysis of the state of the art. *International Journal of Artificial Intelligence in Education (IJAIED)*, 10:98–129. Part II of the Special Issue on Authoring Systems for Intelligent Tutoring Systems (editors: Tom Murray and Stephen Blessing).

Koichi Nagatsuka, Clifford Broni-Bediako, and Masayasu Atsumi. 2021. Pre-training a BERT with curriculum learning by increasing block-size of input text. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, pages 989–996, Held Online. INCOMA Ltd.

Ben Naismith, Phoebe Mulcaire, and Jill Burstein. 2023. Automated evaluation of written discourse coherence using GPT-4. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, pages 394–403, Toronto, Canada. Association for Computational Linguistics.

Melvin R. Novick. 1966. The axioms and principal results of classical test theory. *Journal of Mathematical Psychology*, 3(1):1–18.

John W Oller Jr. 1973. Cloze tests of second language proficiency and what they measure. *Language learning*, 23(1):105–118.

Judith Reitman Olson and Henry H Rueter. 1987. Extracting expertise from experts: Methods for knowledge acquisition. *Expert systems*, 4(3):152–168.

OpenAI. 2023. Gpt-4 technical report.

Subhadarshi Panda, Frank Palma Gomez, Michael Flor, and Alla Rozovskaya. 2022. Automatic generation of distractors for fill-in-the-blank exercises with round-trip neural machine translation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 391–401, Dublin, Ireland. Association for Computational Linguistics.

Amandalynne Paullada, Inioluwa Deborah Raji, Emily M. Bender, Emily Denton, and Alex Hanna. 2021. Data and its (dis)contents: A survey of dataset development and use in machine learning research. *Patterns*, 2(11):100336.

E. Margaret Perkoff, Abhidip Bhattacharyya, Jon Cai, and Jie Cao. 2023. Comparing neural question generation architectures for reading comprehension. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, pages 556–566, Toronto, Canada. Association for Computational Linguistics.

Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J Guibas, and Jascha Sohl-Dickstein. 2015. Deep knowledge tracing. In *Advances in Neural Information Processing Systems*, volume 28, pages 1–9. Curran Associates, Inc.

Ingrid Piller. 2002. Passing for a native speaker: Identity and success in second language learning. *Journal of sociolinguistics*, 6(2):179–208.

Emmanouil Antonios Platanios, Otilia Stretcu, Graham Neubig, Barnabas Poczos, and Tom Mitchell. 2019. Competence-based curriculum learning for neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1162–1172, Minneapolis, Minnesota. Association for Computational Linguistics.

Massimo Poesio, Jon Chamberlain, Udo Kruschwitz, Livio Robaldo, and Luca Ducceschi. 2013. Phrase detectives: Utilizing collective intelligence for internet-scale language resource creation. *ACM Trans. Interact. Intell. Syst.*, 3(1).

Dimitri Prandner and Matthias Forstner. 2022. Are There Enough Open Educational Resources Dealing With Social Science Research Methods? Insights From the D-A-CH Region. *Frontiers in Education*, 7.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):24.

Anna N. Rafferty, Emma Brunskill, Thomas L. Griffiths, and Patrick Shafto. 2016. Faster teaching via pomdp planning. *Cognitive Science*, 40(6):1290–1332.

Dinesh Raghu, Shantanu Agarwal, Sachindra Joshi, and Mausam. 2021. End-to-end learning of flowchart grounded task-oriented dialogs. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4348–4366, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Georg Rasch. 1960. *Probabilistic models for some intelligence and attainment tests.* Danmarks Paedagogiske Institut.

Manav Rathod, Tony Tu, and Katherine Stasaski. 2022. Educational multi-question generation for reading comprehension. In *Proceedings of the 17th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2022)*, pages 216–223, Seattle, Washington. Association for Computational Linguistics.

Evgeniia Razumovskaia, Goran Glavaš, Olga Majewska, Edoardo Ponti, and Ivan Vulić. 2022. Natural language processing for multilingual task-oriented dialogue. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 44–50, Dublin, Ireland. Association for Computational Linguistics.

Monique Reichert, Ulrich Keller, and Romain Martin. 2010. The c-test, the tcf and the cefr: a validation study. *Der C-Test: Beiträge Aus der Aktuellen Forschung. The C-Test: Contributions from Current Research*, pages 205–231.

Rafael Rivera Pastor, Carlota Tarín Quirós, Juan Pablo Villar García, Toni Badia Cardús, and Maite Melero Nogués. 2017. Language equality in the digital age: Towards a human language project. *Study produced for the European Parliament.*

Nathaniel R. Robinson, Perez Ogayo, David R. Mortensen, and Graham Neubig. 2023. Chatgpt mt: Competitive for high- (but not low-) resource languages. *arXiv preprint 2309.07423*.

Frank Rosenblatt. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386–408.

Stuart Russell. 2019. *Human compatible: Artificial intelligence and the problem of control*. Penguin.

Elizabeth Salesky, Kareem Darwish, Mohamed Al-Badrashiny, Mona Diab, and Jan Niehues. 2023. Evaluating multilingual speech translation under realistic conditions with resegmentation and terminology. In *Proceedings of the 20th International Conference on Spoken Language Translation (IWSLT 2023)*, pages 62–78, Toronto, Canada (in-person and online). Association for Computational Linguistics.

Nithya Sambasivan, Shivani Kapania, Hannah Highfill, Diana Akrong, Praveen Paritosh, and Lora M Aroyo. 2021. "Everyone Wants to Do the Model Work, Not the Data Work": Data Cascades in High-Stakes AI. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, pages 1–15, New York, NY, USA. Association for Computing Machinery.

Thomas Scheffer. 1997. Dolmetschen als darstellungsproblem: Eine ethnographische studie zur rolle der dolmetscher in asylanhörungen. *Zeitschrift für Soziologie*, 26(3):159–180.

Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. 2002. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '02, page 253–260, New York, NY, USA. Association for Computing Machinery.

Merel Scholman, Valentina Pyatkin, Frances Yung, Ido Dagan, Reut Tsarfaty, and Vera Demberg. 2022. Design choices in crowdsourcing discourse relation annotations: The effect of worker selection and training. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 2148–2156, Marseille, France. European Language Resources Association.

Alexander Schrijver. 1986. *Theory of linear and integer programming*. John Wiley & Sons.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2020. Green ai. *Commun. ACM*, 63(12):54–63.

Ozan Sener and Silvio Savarese. 2018. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*.

Farida Agus Setiawati, Rizki Nor Amelia, Bambang Sumintono, and Edi Purwanta. 2023. Study item parameters of classical and modern theory of differential aptitude test: Is it comparable? *European Journal of Educational Research*, 12(2):1097–1107.

Burr Settles. 2009. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences.

Burr Settles. 2012. *Active Learning*, volume 18 of *Synthesis Lectures on Artificial Intelligence and Machine Learning*. Morgan & Claypool.

Burr Settles, Chris Brust, Erin Gustafson, Masato Hagiwara, and Nitin Madnani. 2018. Second language acquisition modeling. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 56–65, New Orleans, Louisiana. Association for Computational Linguistics.

Burr Settles, Mark Craven, and Lewis Friedland. 2008. Active Learning with Real Annotation Costs. In *Proceedings of the NIPS Workshop on Cost-Sensitive Learning*, pages 1–10, Vancouver, Canada.

Burr Settles, Geoffrey T. LaFlair, and Masato Hagiwara. 2020. Machine Learning–Driven Language Assessment. *Transactions of the Association for Computational Linguistics*, 8:247–263.

Claude Elwood Shannon. 1948. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423.

Boaz Shmueli, Jan Fell, Soumya Ray, and Lun-Wei Ku. 2021. Beyond fair pay: Ethical implications of NLP crowdsourcing. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3758–3769, Online. Association for Computational Linguistics.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross Anderson. 2024. The curse of recursion: Training on generated data makes models forget.

George Siemens and Phil Long. 2011. Penetrating the fog: Analytics in learning and education. *EDUCAUSE review*, 46(5):30.

Günther Sigott. 1995. The C-Test: Some Factors of Difficulty. *AAA: Arbeiten aus Anglistik und Amerikanistik*, 20(1):43–53.

Günther Sigott. 2006. How fluid is the C-Test construct? In *Der C-Test: Theorie, Empirie, Anwendungen – The C-Test: Theory, Empirical Research, Applications*, Language Testing and Evaluation, pages 139–146. Frankfurt am Main: Peter Lang.

Kathleen Siminyu and Sackey Freshia. 2020. AI4D - African language dataset challenge. In *Proceedings of the The Fourth Widening Natural Language Processing Workshop*, pages 68–77, Seattle, USA. Association for Computational Linguistics.

Adam Skory and Maxine Eskenazi. 2010. Predicting cloze task quality for vocabulary training. In *Proceedings of the NAACL HLT 2010 Fifth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 49–56, Los Angeles, California. Association for Computational Linguistics.

Daniel Smith, Philip Schlaepfer, Katie Major, Mark Dyble, Abigail E Page, James Thompson, Nikhil Chaudhary, Gul Deniz Salali, Ruth Mace, Leonora Astete, et al. 2017. Cooperation and the evolution of hunter-gatherer storytelling. *Nature communications*, 8(1853):1–9.

Eric Alden Smith. 2010. Communication and collective action: language and the evolution of human cooperation. *Evolution and Human Behavior*, 31(4):231–245.

Ard Snijders, Douwe Kiela, and Katerina Margatina. 2023. Investigating multi-source active learning for natural language inference. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2187–2209, Dubrovnik, Croatia. Association for Computational Linguistics.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Ng. 2008. Cheap and Fast – But is it Good? Evaluating Non-Expert Annotations for Natural Language Tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 254–263, Honolulu, Hawaii, USA.

Robert A. Sottilare, Ryan S. Baker, Arthur C. Graesser, and James C. Lester. 2017. Special issue on the generalized intelligent framework for tutoring (gift): Creating a stable and flexible platform for innovations in aied research. *International Journal of Artificial Intelligence in Education.*

Bernard Spolsky. 1969. Reduced redundancy as a language testing tool. In *Second International Congress of Applied Linguistics (Language Testing)*, pages 1–17, Cambridge, England.

Christian Stab and Iryna Gurevych. 2014. Annotating argument components and relations in persuasive essays. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1501–1510, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Lorenz Stangier, Ji-Ung Lee, Yuxi Wang, Marvin Müller, Nicholas Frick, Joachim Metternich, and Iryna Gurevych. 2022. TexPrax: A messaging application for ethical, real-time data collection and annotation. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 9–16, Taipei, Taiwan. Association for Computational Linguistics.

David Stap and Ali Araabi. 2023. ChatGPT is not a good indigenous translator. In *Proceedings of the Workshop on Natural Language Processing for Indigenous Languages of the Americas (AmericasNLP)*, pages 163–167, Toronto, Canada. Association for Computational Linguistics.

Hongjin Su, Jungo Kasai, Chen Henry Wu, Weijia Shi, Tianlu Wang, Jiayi Xin, Rui Zhang, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. 2023. Selective annotation makes language models better few-shot learners. In *The Eleventh International Conference on Learning Representations*, pages 1–24.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. 2020. Dataset Cartography: Mapping and Diagnosing Datasets with Training Dynamics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9275–9293, Online. Association for Computational Linguistics.

Anaïs Tack, Ekaterina Kochmar, Zheng Yuan, Serge Bibauw, and Chris Piech. 2023. The BEA 2023 shared task on generating AI teacher responses in educational dialogues. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building*

Educational Applications (BEA 2023), pages 785–795, Toronto, Canada. Association for Computational Linguistics.

Evelina Tainer. 1988. English Language Proficiency and the Determination of Earnings among Foreign-Born Men. *The Journal of Human Resources*, 23(1):108–122.

Min Tang, Xiaoqiang Luo, and Salim Roukos. 2002. Active Learning for Statistical Natural Language Parsing. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 120–127, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Christopher Tauchmann, Johannes Daxenberger, and Margot Mieskes. 2020. The Influence of Input Data Complexity on Crowdsourcing Quality. In *Proceedings of the 25th International Conference on Intelligent User Interfaces Companion*, pages 71––72, Cagliari, Italy.

Wilson L. Taylor. 1953. "Cloze Procedure": A New Tool for Measuring Readability. *Journalism Bulletin*, 30(4):415–433.

Ahmed Tlili, Boulus Shehata, Michael Agyemang Adarkwah, Aras Bozkurt, Daniel T Hickey, Ronghuai Huang, and Brighter Agyemang. 2023. What if the devil is my guardian angel: Chatgpt as a case study of using chatbots in education. *Smart Learning Environments*, 10(1):15.

Marcos Treviso, Ji-Ung Lee, Tianchu Ji, Betty van Aken, Qingqing Cao, Manuel R. Ciosici, Michael Hassid, Kenneth Heafield, Sara Hooker, Colin Raffel, Pedro H. Martins, André F. T. Martins, Jessica Zosa Forde, Peter Milder, Edwin Simpson, Noam Slonim, Jesse Dodge, Emma Strubell, Niranjan Balasubramanian, Leon Derczynski, Iryna Gurevych, and Roy Schwartz. 2023. Efficient Methods for Natural Language Processing: A Survey. *Transactions of the Association for Computational Linguistics*, 11:826–860.

Huong May Truong. 2016. Integrating learning styles and adaptive e-learning system: Current developments, problems and opportunities. *Computers in Human Behavior*, 55:1185–1193.

Masaki Uto, Yuto Tomikawa, and Ayaka Suzuki. 2023. Difficulty-controllable neural question generation for reading comprehension using item response theory. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, pages 119–129, Toronto, Canada. Association for Computational Linguistics.

Veniamin Veselovsky, Manoel Horta Ribeiro, and Robert West. 2023. Artificial artificial artificial intelligence: Crowd workers widely use large language models for text production tasks.

Lucas Nunes Vieira, Carol O'Sullivan, Xiaochun Zhang, and Minako O'Hagan. 2023. Machine translation in society: insights from UK users. *Language Resources and Evaluation*, 57(2):893–914.

Luis Von Ahn. 2006. Games with a purpose. *Computer*, 39(6):92–94.

Lev Vygotsky. 1978. *Mind in society: The development of higher psychological processes*. Cambridge: Harvard University Press.

Norbert Wiener. 1960. Some moral and technical consequences of automation. *Science*, 131(3410):1355–1358.

Michael Wojatzki, Oren Melamud, and Torsten Zesch. 2016. Bundled gap filling: A new paradigm for unambiguous cloze exercises. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications (BEA)*, pages 172–181, San Diego, CA, USA.

Changrong Xiao, Sean Xin Xu, Kunpeng Zhang, Yufang Wang, and Lei Xia. 2023. Evaluating reading comprehension exercises generated by LLMs: A showcase of ChatGPT in education applications. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, pages 610–625, Toronto, Canada. Association for Computational Linguistics.

Wei Xu. 2019. Toward human-centered ai: a perspective from human-computer interaction. *Interactions*, 26(4):42–46.

Songbai Yan, Kamalika Chaudhuri, and Tara Javidi. 2016. Active learning from imperfect labelers. In *Advances in Neural Information Processing Systems*, volume 29, pages 1–9. Curran Associates, Inc.

Kevin P. Yancey, Geoffrey Laflair, Anthony Verardi, and Jill Burstein. 2023. Rating short L2 essays on the CEFR scale with GPT-4. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, pages 576–584, Toronto, Canada. Association for Computational Linguistics.

Chak Yan Yeung, John Lee, and Benjamin Tsou. 2019. Difficulty-aware distractor generation for gap-fill items. In *Proceedings of the The 17th Annual Workshop of the Australasian Language Technology Association*, pages 159–164, Sydney, Australia. Australasian Language Technology Association.

Nana Yoshimi, Tomoyuki Kajiwara, Satoru Uchida, Yuki Arase, and Takashi Ninomiya. 2023. Distractor generation for fill-in-the-blank exercises by question type. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pages 276–281, Toronto, Canada. Association for Computational Linguistics.

W Quin Yow and Tony Zhao Ming Lim. 2019. Sharing the same languages helps us work better together. *Palgrave Communications*, 5(1):1–11.

Michelle Yuan, Hsuan-Tien Lin, and Jordan Boyd-Graber. 2020. Cold-start Active Learning through Self-supervised Language Modeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7935–7948, Online.

Michelle Yuan, Patrick Xia, Chandler May, Benjamin Van Durme, and Jordan Boyd-Graber. 2022. Adapting Coreference Resolution Models through Active Learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational*

Linguistics (Volume 1: Long Papers), pages 7533–7549, Dublin, Ireland. Association for Computational Linguistics.

Amir Zeldes. 2017. The GUM corpus: Creating multilayer resources in the classroom. *Language Resources and Evaluation*, 51(3):581–612.

Torsten Zesch and Oren Melamud. 2014. Automatic generation of challenging distractors using context-sensitive inference rules. In *Proceedings of the Ninth Workshop on Innovative Use of NLP for Building Educational Applications (BEA)*, pages 143–148, Baltimore, MD, USA.

Chicheng Zhang and Kamalika Chaudhuri. 2015. Active Learning from Weak and Strong Labelers. In *Advances in Neural Information Processing Systems*, pages 703–711, Montréal, Canada.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*, pages 1–43.

Zhisong Zhang, Emma Strubell, and Eduard Hovy. 2022. A survey of active learning for natural language processing. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6166–6190, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Ya Zhou and Can Tao. 2020. Multi-task bert for problem difficulty prediction. In *2020 international conference on communications, information system and computer engineering (cisce)*, pages 213–216. IEEE.

Bowei Zou, Pengfei Li, Liangming Pan, and Ai Ti Aw. 2022. Automatic true/false question generation for educational purpose. In *Proceedings of the 17th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2022)*, pages 61–70, Seattle, Washington. Association for Computational Linguistics.