Differentially Private Methods in Natural Language Processing

Vom Fachbereich Informatik
der Technischen Universität Darmstadt
genehmigte

Dissertation

zur Erlangung des akademischen Grades Dr.-Ing.

vorgelegt von
Timour Igamberdiev
geboren in Woronesch, Russland

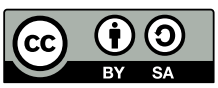
Tag der Einreichung: 01. Juni 2023
Tag der Disputation: 20. Juli 2023

Referenten: Dr. Ivan Habernal, Darmstadt, Germany
Prof. Dr. Iryna Gurevych, Darmstadt, Germany
Prof. Dr. Henning Wachsmuth, Hannover, Germany

Darmstadt 2023
D17

Igamberdiev, Timour: Differentially Private Methods in Natural Language Processing
Darmstadt, Technische Universität Darmstadt
Year thesis published in TUprints: 2023
Day of the viva voce: 20. July 2023
Please cite this document as
URN: urn:nbn:de:tuda-tuprints-244295
URL: http://tuprints.ulb.tu-darmstadt.de/24429

This document is provided by TUprints,
E-Publishing-Service of the TU Darmstadt
http://tuprints.ulb.tu-darmstadt.de
mailto:tuprints@ulb.tu-darmstadt.de



Ehrenwörtliche Erklärung[1]

Hiermit erkläre ich, die vorgelegte Arbeit zur Erlangung des akademischen Grades "Dr.-Ing." mit dem Titel "Differentially Private Methods in Natural Language Processing" selbständig und ausschließlich unter Verwendung der angegebenen Hilfsmittel erstellt zu haben. Ich habe bisher noch keinen Promotionsversuch unternommen.

Darmstadt, den 01. Juni 2023

Timour Igamberdiev

[1] Gemäß §9 Abs. 1 der Promotionsordnung der TU Darmstadt

Wissenschaftlicher Werdegang des Verfassers[2]

09/11 – 05/15	Bachelor of Arts (B.A.) in Linguistics (Honours), Memorial University of Newfoundland
09/16 – 08/18	Master of Arts (M.A.) in Linguistics (Computational Linguistics), Seoul National University
07/20 – heute	Doktorand, Trustworthy Human Language Technologies (TrustHLT) Independent Research Group, Technische Universität Darmstadt

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

Abstract

In today's world, the protection of privacy is increasingly gaining attention, not only among the general public, but also within the fields of machine learning and natural language processing (NLP). An established gold standard for providing a guarantee of privacy protection to all individuals in a dataset is the framework of differential privacy (DP). Intuitively, differential privacy provides a formal theoretical guarantee that the contribution of any individual to some analysis on a dataset is bounded. In other words, no single individual can influence this analysis 'too much'.

While the application of differential privacy to the fields of statistics and machine learning is becoming more widespread, it is still at a relatively early stage in NLP, with many important issues currently unresolved. This includes finding the most favorable methodologies for privatizing textual data that is used to train an NLP system, as well as dealing with the question of privatizing textual data *independent* of an NLP system, releasing it for general analysis, such as for use in a variety of downstream tasks. In this thesis, we address these and other fundamental questions relevant to applying privacy-preserving methods to the field of NLP.

We first present a detailed theoretical background on differential privacy and NLP. We discuss the problem of defining privacy from a philosophical perspective, fundamental concepts in the framework of differential privacy (e.g. the privacy guarantees it provides and how to achieve them), as well as the application of differential privacy to the fields of machine learning and NLP. This is followed by a description of important concepts in the field of NLP, including the structure of a modern NLP system, common tasks of text classification and generation, as well as relevant neural architectures.

We then delve into the primary investigations of this thesis, starting from the privatization of text classification systems. First, we tackle the problem of applying differential privacy to the data structure of graphs used in NLP datasets. Specifically, we demonstrate how to successfully apply the algorithm of differentially private stochastic gradient descent (DP-SGD) to graph convolutional networks, which pose theoretical and practical challenges due to their training characteristics. Next, we move into the territory of more 'standard' NLP models and textual datasets, answering the question of whether a common strategy exists for incorporating DP-SGD in these various settings.

In the second principal set of investigations of this thesis, we focus on the privatization of textual data that is independent of a specific NLP system. In particular, we address this problem from the perspective of privatized text rewriting in the setting of local differential privacy (LDP), in which an entire document is rewritten with differentially private guarantees. We first present our modular framework `DP-Rewrite`, meant to lay down a foundation for the NLP community to solving this task in a transparent and reproducible manner. We then tackle the privatized text rewriting problem itself, proposing the DP-BART model that introduces several techniques which can be applied to a pre-trained BART model, including a novel clipping method, iterative pruning of the model, and further training of inter-

nal representations. Using these techniques, we can drastically reduce the amount of perturbation required to achieve a DP guarantee. We thoroughly examine the feasibility of this approach as a whole, with a focus on the problem of the strict adjacency constraint that is inherent in the LDP setting, which leads to a high amount of perturbation of the original text.

Throughout this thesis, we additionally address several crucial points that are important to keep in mind when applying differential privacy to textual data. First is the question of interpretability, such as what exactly is being privatized in a textual dataset when DP is applied to some analysis on it, as well as the exact details of a proposed DP algorithm and the strength of the privacy guarantee that it provides. Furthermore, it is crucial to be aware of the limitations of proposed methodologies that incorporate DP. This includes computational and memory limitations, as well as the trade-off between the level of privacy that can be provided and the utility of an algorithm, with stronger privacy guarantees expected to more negatively impact utility.

Zusammenfassung

In der heutigen Welt gewinnt der Schutz der Privatsphäre zunehmend an Bedeutung, nicht nur in der Öffentlichkeit, sondern auch in den Bereichen maschinelles Lernen und Natürliche Sprachverarbeitung (Natural Language Processing, NLP). Ein etablierter Goldstandard für die Gewährleistung des Schutzes der Privatsphäre aller Individuen in einem Datensatz ist das Framework von Differential Privacy (DP). Intuitiv bietet Differential Privacy eine formale theoretische Garantie dafür, dass der Beitrag eines jeden Individuums zu einer bestimmten Analyse eines Datensatzes begrenzt ist. Mit anderen Worten, kann keine einzelne Person diese Analyse "zu viel" beeinflussen.

Während die Anwendung von Differential Privacy in den Bereichen Statistik und maschinelles Lernen immer mehr Verbreitung findet, befindet sie sich im NLP-Bereich noch in einer relativ frühen Phase, in der viele wichtige Fragen noch ungelöst sind. Dazu gehört die Suche nach den günstigsten Methoden für die Privatisierung von Textdaten, die zum Trainieren eines NLP-Systems verwendet werden, sowie die Frage der Privatisierung von Textdaten *unabhängig* von einem NLP-System, um sie für allgemeine Analysen freizugeben, z. B. für die Verwendung in einer Vielzahl von nachgelagerten Aufgaben. In dieser Arbeit befassen wir uns mit diesen und anderen grundlegenden Fragen, die für die Anwendung datenschutzfreundlicher Methoden im Bereich von NLP relevant sind.

Zunächst wird ein detaillierter theoretischer Hintergrund zu Differential Privacy und NLP vorgestellt. Wir erörtern das Problem der Definition von Privatsphäre aus einer philosophischen Perspektive, grundlegende Konzepte im Rahmen von Differential Privacy (z. B. die Garantien für die Privatsphäre und wie sie erreicht werden können) sowie die Anwendung von Differential Privacy auf die Bereiche des maschinellen Lernens und NLP. Es folgt eine Beschreibung wichtiger Konzepte im Bereich von NLP, einschließlich der Struktur eines modernen NLP-Systems, allgemeiner Aufgaben der Textklassifikation und -generierung sowie relevanter neuronaler Architekturen.

Anschließend gehen wir auf den ersten Hauptteil dieser Arbeit ein, beginnend mit der Privatisierung von Textklassifikationssystemen. Zunächst befassen wir uns mit dem Problem der Anwendung von Differential Privacy auf die Datenstruktur von Graphen, die in NLP-Datensätzen verwendet werden. Insbesondere zeigen wir, wie man den Algorithmus von Differentially Private Stochastic Gradient Descent (DP-SGD) erfolgreich auf Graphfaltungsnetzwerke anwenden kann, die aufgrund ihrer Trainingseigenschaften theoretische und praktische Herausforderungen darstellen. Als Nächstes befassen wir uns mit 'standardisierten' NLP-Modellen und Textdatensätzen und beantworten die Frage, ob es eine gemeinsame Strategie für die Integration von DP-SGD in diesen verschiedenen Konfigurationen gibt.

Im zweiten Hauptteil dieser Arbeit konzentrieren wir uns auf die Privatisierung von Textdaten, die unabhängig von einem bestimmten NLP-System sind. Insbesondere behandeln wir dieses Problem aus der Perspektive des privatisierten Umschreibens von Text im Rahmen von Local Differential Privacy (LDP), bei der ein ganzes

Dokument mit Differential Privacy Garantien umgeschrieben wird. Wir stellen zunächst unser modulares Framework `DP-Rewrite` vor, das der NLP-Gemeinschaft eine Grundlage bieten soll, um diese Aufgabe auf transparente und reproduzierbare Weise zu lösen. Anschließend gehen wir das Problem der privatisierten Textumschreibung selbst an, indem wir das DP-BART-Modell vorstellen, das mehrere Techniken einführt, die auf ein vortrainiertes BART-Modell angewendet werden können, darunter eine neuartige Clipping-Methode, iteratives Pruning des Modells und weiteres Training interner Repräsentationen. Mit diesen Techniken können wir den Umfang der Störungen, die zur Erreichung einer DP-Garantie erforderlich sind, drastisch reduzieren. Wir untersuchen gründlich die Machbarkeit dieses Ansatzes als Ganzes, wobei wir uns auf das Problem der strengen Adjazenzbeschränkung konzentrieren, die der LDP-Umgebung innewohnt und zu einem hohen Störungsgrad des ursprünglichen Textes führt.

In dieser Arbeit gehen wir zusätzlich auf mehrere entscheidende Punkte ein, die bei der Anwendung von Differential Privacy auf Textdaten beachtet werden müssen. Erstens geht es um die Frage der Interpretierbarkeit, z. B. was genau in einem Textdatensatz privatisiert wird, wenn DP auf eine Analyse darauf angewendet wird, sowie um die genauen Details eines vorgeschlagenen DP-Algorithmus und die Stärke der Datenschutzgarantie, die er bietet. Darüber hinaus ist es von entscheidender Bedeutung, dass man sich der Grenzen der vorgeschlagenen Methoden, die DP einbeziehen, bewusst ist. Dazu gehören Beschränkungen in Bezug auf Rechenleistung und Speicherplatz sowie die Abwägung zwischen dem Grad der Privatsphäre, der gewährleistet werden kann, und dem Nutzen eines Algorithmus, wobei sich stärkere Datenschutzgarantien voraussichtlich negativ auf den Nutzen auswirken.

Acknowledgments

I would like to express my deepest gratitude to everyone who has accompanied me on this incredible and memorable journey. First and foremost I would like to thank Dr. Ivan Habernal for his excellent supervision and very valuable feedback throughout my degree. His guidance helped me to further develop myself and reach my full potential as a researcher. I would also like to thank Prof. Dr. Iryna Gurevych for giving me the opportunity to pursue a Ph.D. at TU Darmstadt and join the very dynamic research environment at the TrustHLT and UKP labs. Furthermore, I would like to thank Prof. Dr. Henning Wachsmuth for devoting the time to review this thesis, along with Dr. Ivan Habernal and Prof. Dr. Iryna Gurevych.

I want to thank my colleagues for their support, company, and insightful feedback during my time at the computer science department: (last name alphabetical order) Luke Bates, Irina Bigoulaeva, Peter Ebert Christensen, Nico Daheim, Nils Dycke, Max Glockner, Lena Held, Sebastian Ochs, Mert Tiftikci, and many more. I greatly appreciate all the thoughtful discussions we have had, during both formal meetings and informal conversations. In addition, I would like to give a special thanks to Luke Bates and former colleague Derek Hommel for their constant friendship and helpful advice as I continued moving along this path of pursuing my degree.

Finally, I would like to express my deepest gratitude to my family, for their love and continuous support over all these years. The achievements that I have attained owe to your enduring guidance and encouragement.

Funding. This work has been supported by the German Federal Ministry of Education and Research, the Hessian Ministry of Higher Education, Research, Science and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE, and by the PrivaLingo research grant (Hessisches Ministerium des Innern und für Sport). Some computations for this research were conducted on the Lichtenberg high performance computer of the TU Darmstadt.

Contents

Chapter 1

Introduction

"Privacy is something you can sell, but you can't buy it back."

— Bob Dylan

1.1 Motivations for Privacy

The notion of privacy is increasingly gaining attention in today's world, both among the general public and within the scientific community. We can consider this from two separate angles: The societal perspective and the research perspective.

We live in a world in which 'data is the new oil',[1] where the value of data for competing businesses and organizations has made it a core asset. It is indispensable for effectively training modern artificial intelligence (AI) systems and provides insights for marketers on targeting their advertisements.

This has subsequently led to extensive data collection and analytics. Every time we enter a search query, access a website, or listen to an audio track on a streaming platform, we are contributing a new data point to some existing dataset. While this may be good for training statistical and machine learning models, it leads to a significant vulnerability of the general public with respect to people's personal data and how it is used.

Apart from the ethical considerations of this problem, there are multiple other concerns for data contributors, depending on the nature of the personal data. For instance, if somebody is contributing online reviews on a business reviewing platform, an individual's negative reviews could lead to business owners taking retaliatory actions, such as lawsuits against them. Another example is the interaction of a user with a dialogue system, in which the individual may reveal information about themselves that would be considered sensitive throughout the interaction.

Arguably just as important for privacy is the research perspective. At the mo-

[1] Phrase widely believed to be coined by UK mathematician Clive Humby in 2006.

ment, it is very difficult to convince data holders to provide certain data. A good example of this are hospital medical records. Institutions such as hospitals are legally required to protect patients' data. For example, in Germany there are strict statutory regulations on medical data, which can be found in the European General Data Protection Regulation (GDPR), the German Federal Data Protection Act (Bundesdatenschutzgesetz),[2] as well as data protection laws for each German state. At the same time, general access to such data would allow for new progress in many fascinating research areas, such as medicine and AI.

As a research community, we are currently able to download a dataset of handwritten digits such as the MNIST database (Bottou et al., 1994), or a dataset of movie reviews (e.g. Maas et al. (2011)) and train a high-performing classifier on handwritten digit recognition or movie review sentiment analysis, respectively; however, we cannot easily access something like a cancer dataset (e.g. medical images, medical notes) and try to more successfully detect whether an individual has cancer or other diseases. In this way, developing privacy-preserving technologies would open up a wide new world of research possibilities that could potentially have great benefits for humanity. We can therefore see that *privacy leads to open science*.

So given these benefits and urgent need for privacy-preserving methodologies, what can we do as a research community? One naive option that immediately may come to mind is that of anonymization, where we basically remove any identifying information from a given data point. For instance, given a doctor's medical note, we could remove any mention of the patient, any background information about them such as age or gender, and so forth. As has been extensively demonstrated, however, this approach has fundamental flaws (Sweeney, 1997; Narayanan and Shmatikov, 2008; Homer et al., 2008). One major reason for this is that of linkage attacks, in which information about anonymized individuals is re-identified by means of linking the anonymized data with background information. This was notably demonstrated by Narayanan and Shmatikov (2008) on an anonymized dataset released by the streaming service Netflix, linking movie ratings and dates with similar public data available on the Internet Movie Database (IMDb).

It is therefore evident that anonymization methods are unsatisfactory for providing privacy for individuals. While there have been some more formal techniques proposed with the goal of providing stricter privacy guarantees, such as *k-anonymity*, (Samarati and Sweeney, 1998; Sweeney, 2002), these are still vulnerable to privacy violations, including through linkage with background information. It was not until the emergence of *differential privacy* (DP) (Dwork et al., 2006b; Dwork and Roth, 2013) that we obtained a method with rigorous, quantifiable privacy guarantees, neutralizing privacy violations such as the above linkage attacks and providing a strict measure of the degree of an algorithm's privacy protections. On a very intuitive level, *differential privacy is a property of an algorithm, in which the output of the algorithm cannot change by more than a very specific amount when one data point is added, removed, or altered from a dataset.* In other words, no single individ-

[2] https://www.gesetze-im-internet.de/bdsg_2018/BJNR209710017.html#BJNR209710017BJNG000100000

ual can contribute to the analysis of a dataset 'too much', thus not standing out from the other contributors to the dataset. This is achieved through the incorporation of *randomness* into the analysis, perturbing either its output, or the input data itself. The exact degree of an individual's contribution is determined by the parameter ε, or the *privacy budget*, which dictates how much perturbation is introduced to the algorithm. The lower this ε value, the stronger the privacy guarantee.

Since its inception, differential privacy has sparked an entire field of research, being considered the 'gold standard' in privacy-preservation. In 2016 it has won the Test of Time award and is currently used by many large companies, such as Microsoft, Apple, and Google, as well as by the U.S. Census Bureau since 2020. Although it is becoming more widespread within the field of machine learning as well (e.g. Abadi et al. (2016b); Papernot et al. (2017)), it is still at a relatively early stage in Natural Language Processing (NLP), with some important issues that are currently unresolved. Among these, two considerable types of challenges stand out. The first is finding the most favorable methodology to privatizing the process of text classification in NLP systems, including how to deal with various 'non-standard' data types in NLP (e.g. graphs), as well as strategies to employ for different NLP tasks (e.g. sentiment analysis, named-entity recognition (NER), question answering, and so forth). Furthermore, and perhaps far more difficult, is the question of privatizing textual data itself (as opposed to NLP *systems* above), including what it even means to guarantee privacy for language data. An essential component of finding a solution to these questions is dealing with the *privacy/utility trade-off*, since incorporating differential privacy into an algorithm inevitably results in a utility loss for the algorithm due to the introduced perturbation. The more privacy is introduced, the more perturbation, and therefore the greater the expected loss in performance. It is the goal of this thesis to address and move closer to resolving these open research questions, outlined in more detail below.

1.2 Research Questions

1.2.1 Primary research questions

The current thesis investigates the following research questions in detail. These can be broadly divided into two main categories: (a) The privatization of text classification models (**RQ1** and **RQ2**), as well as (b) the privatization of textual data itself (**RQ3**).

RQ1 How can we privatize text classification models that operate on graph datasets? This relates to the more general question of how to privatize models for 'non-standard' data in NLP, such as those that utilize graph structures as opposed to linear sequences of text tokens. In dealing with graphs, a variety of new problems emerge, since not only do individual node or edge features of graphs need to be protected, but also the graph structure itself, which can reveal sensitive information about the relationships of graph entities. We demonstrate that applying the most common machine learning model privatization method, the DP-SGD algorithm (Abadi et al., 2016b), cannot be straightforwardly applied to graph

convolutional networks (GCNs) due to the requirements of DP-SGD to split up data into i.i.d. sub-samples called *lots*. We therefore adapt DP-SGD to the graph setting by preparing a graph partitioning algorithm, recovering a lot of dropped performance in the DP setting, in comparison to not splitting the graph. This research question is investigated in Chapter 3.

RQ2 Is there a systematic strategy that can be applied to NLP text classification tasks in the differentially private setting? Moving into the domain of more 'standard' NLP models and tasks, we investigate the application of differential privacy as a paradigm across a variety of common NLP tasks and models. While DP-SGD has been used in language modeling (McMahan et al., 2018; Hoory et al., 2021), the NLP community lacks a thorough understanding of its usability in the NLP domain in general, with existing research on the suitability of DP-SGD for various NLP tasks remaining largely inconclusive. We are interested first in which models and training strategies provide the best trade-off between privacy and performance across tasks. Second, we investigate exactly how increasing the privacy requirements can hurt the performance for a given model and task. As in the case of our GCN investigation for RQ1, we employ the DP-SGD algorithm for privatizing our NLP models. We thus address the various challenges of each NLP task in privacy-preserving learning and explore whether there is a 'one size fits all' privatization strategy in the field of NLP. This problem is tackled in Chapter 4.

RQ3 How can we successfully privatize textual data, independent from a specific NLP system? More concretely for our investigations, this research question can also be reworded as: *To what extent is local differential privacy (LDP) possible for textual data?* Local differential privacy, outlined in more detail in Section 2.2.1 of Chapter 2, is a specific form of differential privacy, in which the *data itself is perturbed*, as opposed to the machine learning algorithm acting upon the data, as in RQ1 and RQ2. One particular setup of achieving this is through differentially private text rewriting, in which a given piece of text (e.g. an entire document) is rewritten with differentially private guarantees, removing personal information associated with the individual contributing this text. For instance, given a document "I would like to fly from Denver to Los Angeles this Thursday", the system may rewrite it as "Show me flights to cities in California this week". If one is training a model on intent classification for airline travel inquiry systems, either document would be a useful data point. In this way, we avoid using the original text that has uniquely identifiable qualities of a specific author, and instead create a privatized 'synthetic' example.

Overall, this research question can largely be split up into two related problems. The first is an issue currently present in the field of text privatization, where the community lacks reproducibility and transparency in the task. Only a few recent works have touched upon this challenging topic; however, these studies either have been found to have formal flaws (Habernal, 2021), or have not published their source codes, with the community having no means of performing empirical checks to validate their privacy-preserving claims. This lack of transparency and reproducibility is therefore the main obstacle to the accountability of DP text rewriting

systems. We address this issue by developing `DP-Rewrite`, an open, modular, and highly customizable framework for differentially private text rewriting, in order for the community to gain further insight into the utility and potential pitfalls of such systems. Our hypothesis is that by integrating various downstream datasets, models, pre-training procedures, and evaluation metrics into one software package, we improve the transparency, accountability, and reproducibility of research in differentially private text rewriting. We investigate this first part of RQ3 in Chapter 5.

The second part of this research question is one of designing an effective LDP text rewriting model that improves the privacy/utility trade-off in comparison to previous methods. This is in fact a very challenging task, since a lot of perturbation of the original text is required to achieve any reasonable privacy guarantees, which leads to poor downstream utility. We therefore address these issues by proposing **DP-BART**, which is an LDP text rewriting system that improves upon existing baselines and consists of several techniques that can be directly applied to a pre-trained BART model (Lewis et al., 2020a). One major issue that we discuss is the *strict text adjacency constraint* of applying LDP to textual data, in which *any two texts* need to have a degree of indistinguishability for a useful privacy guarantee in the LDP setting, leading to the large amount of perturbation that needs to be applied to the text. This second part of RQ3 is presented in Chapter 6.

1.2.2 Additional research points addressed

Apart from the above three research questions, there are several additional important points that are addressed throughout the various chapters of this thesis and are very relevant to all the investigations presented. These can be divided into two primary categories.

The first category comprises questions of **interpretability**. When we apply differential privacy to a given textual data point, what exactly are we privatizing? What does it even mean to 'privatize' text? Additionally, what exactly does an ε privacy budget mean, given an NLP model or a textual document? With regards to this, we note several pitfalls that can occur in a research investigation utilizing the framework of differential privacy that need to be prevented, including (1) the privacy guarantees of a proposed DP algorithm do not hold up, (2) the interpretation of ε is not clearly defined, and (3) the 'unit of privatization', i.e. what is being privatized, is not clearly defined. We discuss this throughout the thesis, especially in Chapters 2 and 6.

In addition, it is very important to address the **limitations** of our methodologies, including the application of DP to NLP systems in general. Apart from issues of performance loss that we tackle in Chapters 3-6, this also includes computational and memory hindrances that are introduced when using an algorithm such as DP-SGD, or preparing an LDP text rewriting system. For example, the DP-BART model that we present in Chapter 6 is designed to avoid pre-training a large-scale Transformer (Vaswani et al., 2017) model from scratch, which would otherwise require a very large amount of computational resources.

1.3 Contributions

We summarize the most important contributions of this thesis with respect to the above research questions as follows:

RQ1 How can we privatize text classification models that operate on graph datasets?

- We propose a methodology for applying differentially private stochastic gradient descent and its variants to graph convolutional network models, allowing to maintain strict privacy guarantees and performance. Our approach consists of applying an easy-to-implement graph splitting algorithm to GCNs in the DP setting, partitioning a graph into subgraphs while avoiding additional queries on the original data. We adapt DP-SGD and the differentially private version of the Adam optimizer (Kingma and Ba, 2014) DP-Adam to GCNs.

- We conduct experiments on five datasets in two languages (English and Slovak), covering a variety of NLP tasks, including research article classification in citation networks, Reddit post classification, and user interest classification in social networks, where the latter two tasks deal with data that carries inherently sensitive information, reaffirming the need for privacy-preserving models.

- We show that DP training can be applied to the case of GCNs, with graph splitting and proper optimization recovering a lot of the dropped performance that stems from the addition of DP noise.

- We also show that more sophisticated text representations further mitigate this performance drop, resulting in a relative performance of 90% of the non-private models, while keeping strict privacy ($\varepsilon = 1.0$ when incorporating graph splitting).

- To the best of our knowledge, this was the first study that brought differentially private gradient-based training to graph neural networks.

RQ2 Is there a systematic strategy that can be applied to NLP text classification tasks in the differentially private setting?

- We provide an extensive analysis of different privacy-preserving strategies on seven downstream datasets in five common NLP tasks with varying complexity, using varying privacy regimes. We use modern neural models based on BERT (Devlin et al., 2019) and XtremeDistilTransformers (Mukherjee et al., 2021) architectures.

- We show that, unlike standard non-private approaches to solving NLP tasks, where bigger is usually better, privacy-preserving strategies do not exhibit a winning pattern, and each task and privacy regime requires a special treatment to achieve adequate performance. Our main contribution is thus to help the

NLP community better understand the various challenges that each task poses to privacy-preserving learning.

RQ3 How can we successfully privatize textual data, independent from a specific NLP system?

- With respect to the first part of RQ3, dealing with the problems of reproducibility and transparency, our contributions are twofold. First, we present `DP-Rewrite`, an open-source framework for differentially private text rewriting experiments. It includes a correct reimplementation of the ADePT text rewriting system (Krishna et al., 2021) as a baseline, integrates pre-training on several datasets, and allows us to easily perform downstream experiments with varying privacy guarantees by adjusting the privacy budget ε. Second, `DP-Rewrite` allows us to easily detect another privacy leak in the approach proposed in ADePT, namely in the pre-training strategy of the autoencoder, with the system memorizing the input data. We demonstrate this in detail as a use-case of `DP-Rewrite`.

- For the second part of RQ3, designing an effective LDP text rewriting model, we can divide our contributions into three main points. First, we present our **DP-BART** system and its related methodologies, aimed at reducing DP noise and reaching a better privacy/utility trade-off. For comparison, we use a reimplementation of the ADePT model mentioned above, which is the current primary baseline for this task. Second, we run experiments to investigate the privacy/utility trade-off of these models, using five unique datasets that gradually increase in size, evaluating rewritten texts on downstream text classification tasks. Finally, we thoroughly examine the feasibility of the LDP text rewriting setting, investigating issues of the high noise requirement due to the strict text adjacency constraint, trade-offs between privacy and dataset size, what exactly is the object of privatization, required computational resources, as well as limitations of the approach as a whole and possible alternatives.

1.4 Publication Record

The various sections of this thesis have been published at international peer-reviewed conferences, as well as on the arXiv pre-print server. These publications are partly reused and quoted verbatim throughout the various sections of this thesis. We present these publications in the following list, with the corresponding dissertation chapter from which verbatim quotes from this publication are included. In addition to the below list, some parts of the introduction from each corresponding publication are used in the current chapter's list of research questions (Section 1.2.1) and contributions (Section 1.3).

- *Privacy-Preserving Graph Convolutional Networks for Text Classification* (Igamberdiev and Habernal, 2022). In this publication, we demonstrate a methodology to adapt the DP-SGD algorithm to the case of graph convolutional networks, demonstrating how we can achieve a better privacy/utility trade-off

when incorporating a graph splitting algorithm. This work forms the basis of Chapter 3, in which parts of it are quoted verbatim.

- *One size does not fit all: Investigating strategies for differentially-private learning across NLP tasks* (Senge et al., 2022). This is a joint first-author publication in which we investigate strategies for applying differential privacy on seven downstream NLP datasets and five different NLP tasks, using a variety of privacy budgets. We show that there is no winning pattern with respect to applying privacy preservation in the NLP setting, with each specific task and dataset requiring special treatment to obtain a good performance. This study contributes to the core of Chapter 4, in which it is paraphrased.

- *DP-Rewrite: Towards Reproducibility and Transparency in Differentially Private Text Rewriting* (Igamberdiev et al., 2022). In this work, we present our `DP-Rewrite` software, an open, modular and easily extensible framework which is meant to address the transparency and reproducibility problem of differentially private text rewriting systems. Furthermore, we provide a case study on the ADePT DP text rewriting system and demonstrate a privacy leak in its pre-training strategy. This publication forms the core of Chapter 5, in which it is quoted verbatim.

- *DP-BART for Privatized Text Rewriting under Local Differential Privacy* (Igamberdiev and Habernal, 2023). The final publication presents a novel LDP text rewriting system, DP-BART, in which we improve upon previous systems with respect to the privacy/utility trade-off, while retaining the low-resource setting. We run experiments on five unique datasets of gradually increasing size and evaluate rewritten texts on various downstream text classification tasks. Importantly, we also discuss the feasibility of the LDP text rewriting setting, in addition to addressing several other relevant problems, such as the interpretability of this type of private text rewriting method. This publication contributes significantly to Chapter 6, as well as to a few parts of Section 2.2 in Chapter 2, which quote it verbatim.

1.5 Overview of Thesis

The current thesis can be divided into three main parts and largely follows the order of publications that is presented in Section 1.4, as well as the contributions described in Section 1.3.

Part 1: Chapters 1 and 2 form a general **background** to the thesis. In Chapter 2, we provide a discussion on what is privacy from a general perspective, followed by a comprehensive introduction to differential privacy. This includes explanations of *pure* and *approximate* differential privacy, with a description of relevant concepts such as the *privacy budget*, *query*, and *sensitivity*. Additional concepts that are addressed in this section include common DP mechanisms such as the Laplace and Gaussian mechanisms, the distinction between global vs. local DP, as well as important algorithmic properties of differential privacy. We then provide a pedagogical

example of a differentially private mechanism in the global DP setting, as well as a description of the *randomized response* technique (Warner, 1965), which forms a pedagogical basis of a locally differentially private mechanism. After this, we describe differential privacy in the machine learning setting, going over in detail the most common DP algorithm for training neural network models, *DP-SGD* (Abadi et al., 2016b), as well as the *Moments Accountant* that is useful for the efficient composition of multiple DP mechanisms. This is then followed up with a discussion of differential privacy in the NLP setting. Finally, we end this section with an explanation of relevant NLP concepts, including the standard structure of a modern NLP system, common NLP tasks of text classification and text generation, as well as relevant neural architectures, such as Transformer models (Vaswani et al., 2017).

Part 2: Chapters 3 and 4 tackle the first problem of **privatizing the process of text classification**. In Chapter 3, we present our approach for applying differential privacy to the 'non-standard' NLP data structure of graphs. We specifically demonstrate how to apply the DP-SGD algorithm to graph convolutional networks. Having investigated the scenario of NLP in the graph setting, we then move back to the 'standard' models and textual datasets that are commonly found in NLP. We present our study on privatizing these models and datasets in Chapter 4, answering the question of whether a common strategy exists for incorporating DP-SGD to these various settings.

Part 3: Chapters 5 and 6 address the second problem of **privatizing textual data, independent of a specific NLP system**. Chapter 5 presents our `DP-Rewrite` framework and its accompanying case-study. This is meant to lay down a foundation for the community to work on solving this problem in a transparent and reproducible manner. Chapter 6 then tackles the text privatization problem itself, describing the proposed DP-BART text rewriting model, as well as addressing limitations of the approach as a whole and other relevant questions, such as the impact of the domain of public training texts on the model, and interpretability of the approach.

Finally, Chapter 7 **summarizes** this thesis and provides suggestions for **future directions** and concluding remarks.

Chapter 2

Theoretical background

2.1 What is Privacy?

There are a variety of ways to define what privacy actually is, with no clear consensus on a single definition. For the purposes of this thesis, we use the formal privacy model outlined in the framework of Differential Privacy (DP), described in Section 2.2. This model provides a guarantee that, if an individual contributes their data to a private database, they will not be negatively impacted by the results of some analysis on that database, any more than if they had not contributed their data in the first place (Dwork and Roth, 2013). Thus, the degree of harm that may come to any individual is not affected by whether they gave up their data or not, regardless of other studies or information from additional sources. As an example from Dwork (2011), an individual who is a smoker provides their data to a database that is used to analyze the connections between smoking and cancer. It turns out that a key finding of this analysis is that smoking does indeed cause cancer. As a result of this, the insurance premiums of the individual that participated end up rising. While this negatively impacts the individual, it was not due to their participation in the original dataset, but rather by the findings of the analysis on the dataset itself. The guarantee of differential privacy is that any harm that the individual may face can only be as a result of this analysis, and not due to user participation in it.

A particularly appealing aspect of working in the framework of differential privacy is that we can provide *formal and quantifiable* privacy guarantees. We can show a direct comparison among a variety of techniques and concretely measure for each of them the maximum privacy loss that any individual may incur. In this way, we can investigate the trade-off between privacy and utility of algorithms, trying to reach as strong of a privacy guarantee as possible, with the smallest drops in utility of the algorithm. Notably, the guarantee provided by differential privacy is an *information-theoretic* guarantee, contrasting with guarantees provided in the field of security that are based on computational complexity, where an adversary is unable to solve a given problem in a computationally-efficient manner. In our case, no matter the adversary's computational capabilities, it is impossible to break

the provided guarantee, even with unlimited computational resources and time. Before delving into the specifics of this framework in more detail, we first present an overview of ways in which privacy has been defined throughout the literature, in order to provide a multifaceted perspective on the concept and link it to our own approaches to privacy.

2.1.1 Theories of Privacy

There are different ways of categorizing definitions and theories of privacy. One such organization is by Tavani (2007), dividing theories of privacy into four categories: Non-intrusion, seclusion, control and limitation. The first of these, **non-intrusion**, can be traced to a fundamental publication on privacy by Brandeis and Warren (1890). In it, privacy is seen as the "right to be left alone", where the individual is free from intrusion. Privacy is connected with the fundamental rights of an individual with respect to their person and property.

Next is the **seclusion** theory of privacy, in which privacy is likened to the state of being alone. As described by Gavison (1980), privacy is related to "our accessibility to others", in terms of us being known to others, the degree to which others direct their attention towards us, as well as whether others have physical access to us. This can in turn help us identify exactly where we have incurred a privacy loss, if the satisfaction of conditions such as the above is compromised.

In contrast, the **information control** theory of privacy treats the concept from the perspective of having control over information relating to oneself (e.g. Westin (1968); Rachels (1975); Fried (1990)). Here, an individual that enjoys privacy is one who is in control over granting or denying information to others about themselves. One can therefore choose the manner and extent to which others know about them. This contrasts with the above two theories of non-intrusion and seclusion, both of which do not include the active role of an individual with respect to their personal information.

Finally, the **limitation** theory of privacy defines the notion as the limitation or restriction of access to information about oneself. For instance, Parent (1983) describes privacy as a condition in which others do not have "undocumented personal knowledge" about oneself. Under such a theory, privacy can be defined as a specific sphere associated with an individual, to which others have limited access.

We can see quite a bit of overlap among these different theories. This is especially notable among the first two theories, as well as the last two. Both the non-intrusion and seclusion views of privacy treat the concept as one of others having access to an individual. The former approaches the concept more from the perspective of an individual's fundamental rights, while the latter from a descriptive perspective, outlining privacy as a state. The information control and limitation theories both deal with limiting information to others, with the former highlighting the individual's control over this process, while the latter again describing the state of this limitation.

Despite such a classification as the one above, individual viewpoints in the lit-

erature do not necessarily fall into one specific category, with for instance Gavison (1980) clearly having consistent views with both the seclusion and the limitation theories. Additionally, all of the above theories have their own shortcomings, as pointed out by Tavani (2007). For instance, the non-intrusion theory of privacy seems to have confusions with the notion of **liberty**, in which the individual loses the ability to freely act as they desire when intruded upon. An example of where the two concepts differ is when one's privacy is intruded upon without one's knowledge. If somebody's private conversations are read without their knowledge, this is a clear privacy violation, but there are no restrictions with respect to liberty such as freedom of expression and thought.

For the seclusion theory, there may be confusions with the idea of **solitude**. Tavani (2007) points out the example that if someone is stranded on an uninhabited island, then under this definition they enjoy "perfect privacy". Similarly, the limitation theory may be confused with **secrecy**, which certainly overlaps with privacy, but is not equivalent. This point is discussed in detail in Thompson (2001), who notes that there are clear instances in which private matters can be general knowledge, but not necessarily secret. He draws the example of an individual's religious practices, which are a private matter to them, but nevertheless may be generally known. In such a case, that individual's privacy may be violated by using this information in an inappropriate way, for instance in discriminating against them. In contrast, secret information simply refers to information that is not known by at least one person.

Finally, it seems that the information control theory of privacy is one of the most popular today (e.g. strongly advocated by Moore (2008), Tavani (2007) opting for a unified privacy theory of control and limitation, and so forth). Moore (2008) relates the control over one's privacy to that of intellectual property, in contrast to that of physical property, such as physical goods or objects. Nevertheless, Tavani (2007) points out a limitation of control-based theories, noting some overlap with the concept of **autonomy**. Suppose that an individual reveals all of their personal information to others, being fully in control of this process and doing so willingly. Under the control theory, this individual would still be enjoying privacy, while our intuition clearly would say otherwise.

In addition to the above categorization of privacy theories, we can draw further distinctions. One clear distinction is between **normative** and **non-normative** (i.e. **descriptive**) views of privacy (Moore, 2008). In the normative case, one is referring to certain moral obligations and ethical concerns. For example, the limitation perspective can be seen as non-normative, since it describes privacy as a condition (e.g. (Parent, 1983)) in which others have limited access to a sphere of information about the individual. In contrast, the non-intrusion view of privacy treats it more from the normative account, with privacy seen as a fundamental right that must not be intruded upon.

Another type of categorization is between **reductionist** and **non-reductionist** views of privacy. In the former account, privacy is considered to be derived from other concepts, such as the right to life or property. In this case, there is therefore

no single concept of 'privacy', but rather a combination of fundamental ideas that are brought together. Thus, if the privacy of an individual is intruded upon, this is really a collection of more underlying violations. In contrast, the non-reductionist view considers privacy as a distinct, fundamental concept, alongside other essential rights.

We can therefore see that, in the legal and philosophical literature, there is no clear consensus of what privacy is, with strong overlaps between the notions of privacy and other related concepts, such as autonomy and secrecy. For this reason, it is crucial to have a formal and rigorous definition of privacy that we can apply in the fields of natural language processing, machine learning, and statistics in general. In this way, we would be able to clearly compare the privacy claims of different algorithms and how they relate to their performance.

There are significant risks of the misuse of people's personal data. At the same time, there are immense research benefits from being able to make sensitive data available for the scientific community, as outlined in Section 1.1. The framework of differential privacy satisfies these desired properties, providing us with the tools that we need for quantifiably privatizing our statistical models.

2.2 Differential Privacy

2.2.1 General Concepts

Differential privacy (DP) was originally proposed by Dwork et al. (2006b) and outlined in more detail in Dwork and Roth (2013). It is a formal, mathematical guarantee that the output of some analysis on a given dataset is nearly indistinguishable when any one data point is modified. In other words, no individual can stand out as a result of this analysis, preserving their privacy.

Neighboring datasets

To define this more formally, we first outline the notion of *neighboring datasets*. We start with a dataset D, which consists of n individuals, $x_1, \ldots, x_n$. We can then define what it means for two datasets to be *neighboring* as follows.

Definition 2.2.1. *Two datasets D and D' are considered* ***neighboring*** *if they differ in at most one record, i.e., one individual's data point. This means that either $D' = D \pm 1$, or $D' = D$ with the i-th data point replaced.*

When considering a dataset as a database of individuals, with rows corresponding to particular individuals and columns corresponding to features of data associated with those individuals, we can see that two datasets, D and D' are neighboring if they differ in one row, i.e. $||D - D'||_1 \leq 1$.

Query and randomized mechanism

Next, in DP we typically refer to a *query* on a dataset, as defined below.

Definition 2.2.2. *A **query** is a function $f : D \to \mathbb{R}^k$ that we evaluate on a dataset D.*

This can range from very simple queries, such as taking the average length of a document, to more complex queries, e.g. a deep learning model predicting the sentiment of a document. In simple terms, the query is the 'question' that the analyst is asking about the data.

In order to provide a formal privacy guarantee, we add *randomness* to this query by perturbing $f(D)$. We refer to this randomized function as a *randomized mechanism* $\mathcal{M}(D; f)$.

Formal definition of DP

The formal definition of differential privacy can now be described as follows.

Definition 2.2.3. *For $\varepsilon \geq 0$, a mechanism $\mathcal{M}$ is ε-differentially private if, for all $S \subseteq Range(\mathcal{M})$, and for any two neighboring datasets D and D', the following holds true:*

$$\Pr[\mathcal{M}(D) \in S] \leq e^{\varepsilon} \Pr[\mathcal{M}(D') \in S] \tag{2.1}$$

To describe the above guarantee more informally: For all subsets in the range of a mechanism, i.e. for all events that could possibly happen, the probability that the event happens with dataset D is going to be 'close' to the probability of that event when dataset D' is fed into the mechanism. This closeness is formally treated as a multiplicative guarantee, hence the probability of every event in S is preserved up to this e^{ε}.

Privacy budget (ε)

Importantly, ε is the *privacy budget*. The lower it is, the more private the mechanism is, since the two output distributions of $\mathcal{M}(D)$ and $\mathcal{M}(D')$ are constrained to be more similar. ε is thus the bound on the amount of privacy leakage that is allowed to occur. We approach a scenario of ideal privacy, but impaired utility, when $\varepsilon \to 0$. In the extreme case when $\varepsilon = 0$, the output of $\mathcal{M}$ is not dependent on D at all (e.g. being fully random), and therefore useless for extracting any information from the dataset. Conversely, in the case of $\varepsilon \to \infty$, we approach the original non-DP setting. In this case, there is no bound on the output distribution of $\mathcal{M}$, which offers no formal privacy guarantees, but may be good for utility. It is important to note that ε is an exponent in Definition 2.1. This means that, as we linearly increase it, the privacy guarantee deteriorates exponentially.

An important and recurring problem in DP is the question of how to find the best trade-off between privacy and utility. The goal is to reach the best possible privacy guarantee, while at the same time not giving up too much utility. This leads to the question of what is a good ε value to aim for in a DP mechanism. While the 'right' value of ε may be dependent on the specific query that is computed and

nature of the data (Lee and Clifton, 2011), as we outline throughout this thesis, a good rule of thumb for an ε value that provides a strong privacy guarantee is the randomized response technique with fair coin flips, with $\varepsilon \approx 1.1$. We describe randomized response in detail in Section 2.2.3 and provide a further discussion on selecting a suitable ε value in Section 6.7 of Chapter 6.

Privacy loss random variable

Another way of looking at the above guarantee in Definition 2.2.3 is to define the *privacy loss random variable*. For a given output $y \sim \mathcal{M}(D)$:

$$\mathcal{L}^{(y)}_{\mathcal{M}(D)||\mathcal{M}(D')} = \ln\left(\frac{\Pr[\mathcal{M}(D) = y]}{\Pr[\mathcal{M}(D') = y]}\right) \tag{2.2}$$

Eqn. 2.2 can be obtained by rearranging Eqn. 2.1 and focusing on a particular observation of $\mathcal{M}$. It defines the *privacy loss* that is incurred through the observation of y, which may be more or less likely to be output when using database D, than when using D'. In other words, the probability mass of $y \sim \mathcal{M}(D)$ may be larger or smaller, respectively, than the mass of $y \sim \mathcal{M}(D')$. It is this loss that is bounded in the DP setting, for all neighboring datasets D and D', and all possible outputs y:

$$\max_{\forall y} \left|\mathcal{L}^{(y)}_{\mathcal{M}(D)||\mathcal{M}(D')}\right| \leq \varepsilon \tag{2.3}$$

ℓ_1-sensitivity

The question then arises: How do we actually achieve a particular ε-DP guarantee for our desired query f? For this, we first define the notion of *sensitivity*.

Definition 2.2.4. *Given a function $f : D^n \to \mathbb{R}^k$ that takes in a dataset D and outputs a vector of dimension k, the ℓ_1-**sensitivity** of the function is as follows:*

$$\Delta_1^{(f)} = \max_{D,D'} ||f(D) - f(D')||_1, \tag{2.4}$$

for all neighboring datasets D, D'.

In other words, the sensitivity is the maximum amount by which the output of algorithm f can change, when we alter one data point. The larger this sensitivity is, the more perturbation we would need to introduce into f in order to obscure the presence of a given individual. In the differentially private setting, we thus add random noise to f based on this sensitivity.

Laplace mechanism

The specific type of random noise varies depending on the particular DP mechanism that is used. One very standard mechanism in DP is the *Laplace Mechanism*. The Laplace Distribution is a continuous probability distribution with location parameter μ and scale parameter b, as shown in Figure 2.1. It can be seen as a two-sided exponential distribution and is defined as follows:

$$f(x|\mu, b) = \frac{1}{2b}\exp\left(-\frac{|x - \mu|}{b}\right) \tag{2.5}$$

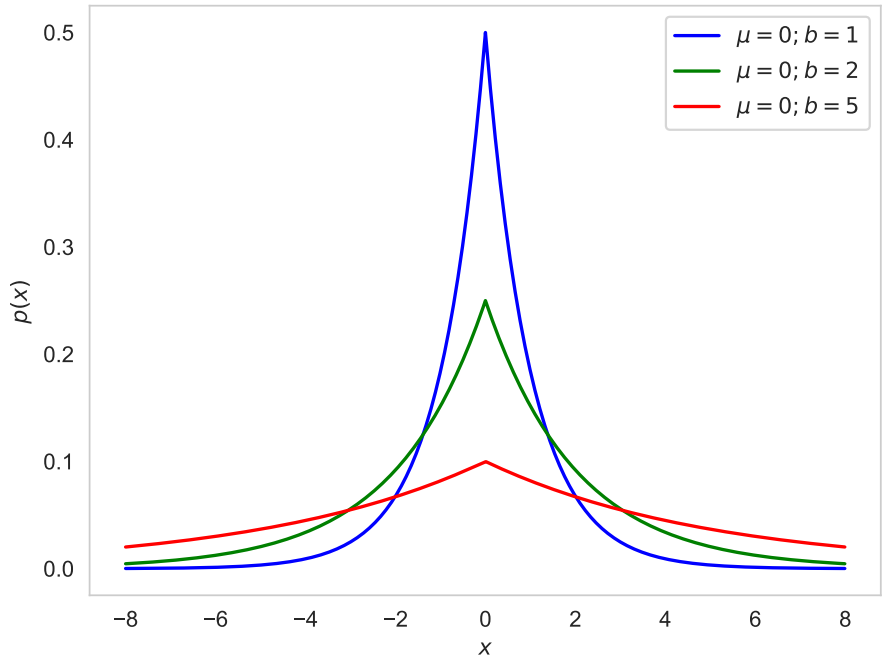


Figure 2.1: Three instantiations of the Laplace distribution, all centered at 0 ($\mu = 0$), with varying scale parameters b. As b is decreased, we can see more of the probability mass around μ (e.g. blue curve at $b = 1$). As the b parameter is increased, the probability mass becomes more spread out, with a far lower peak (e.g. red curve at $b = 5$). The Laplace distribution is symmetrical around its mean and unimodal, meaning that there is only a single peak, at μ.

For our purposes, we will always center the distribution at $\mu = 0$.

To achieve a particular ε-DP guarantee, we will add noise from the Laplace distribution, $\text{Lap}(b)$, to each coordinate of our query f, proportional to the ℓ_1-sensitivity of f. We outline this more formally in Definition 2.2.5.

Definition 2.2.5. *The Laplace mechanism is defined for a given function $f : D^n \to \mathbb{R}^k$ as follows:*

$$\mathcal{M}_L(D, f(\cdot), \varepsilon) = f(D) + (Y_1, \ldots, Y_k), \tag{2.6}$$

where each random variable Y_i is drawn i.i.d. from $\text{Lap}(\Delta_1^{(f)}/\varepsilon)$.

As a result of this process, we can thus achieve a desired ε-DP guarantee for our original query f.

Theorem 2.2.1. *The Laplace mechanism preserves ε-differential privacy.*

Proof. We refer to the proof of Theorem 3.6 in Dwork and Roth (2013). □

Properties of DP

Differential privacy has several very useful properties that make it suitable for statistical data analysis in the private setting. The first of these is that it is *closed under post-processing*. Once we have obtained an output from a differentially private algorithm, we can never recover the original data, more than the ε guarantee allows us, without actually having access to the private data. We therefore cannot make the output of a private algorithm less differentially private. More formally, if a randomized mechanism $\mathcal{M} : D^n \to Y$ is ε-differentially private, and $\mathcal{F} : Y \to Z$ is another randomized mapping, then $\mathcal{F} \circ \mathcal{M}$ is also ε-differentially private. We refer to Dwork and Roth (2013), proposition 2.1 for the proof.

Next, differential privacy allows for the *composition* of private mechanisms. This answers the question of what happens when we want to ask multiple queries on a given dataset. Particularly, this will be important in the case of applying differential privacy to machine learning, outlined in more detail in Section 2.2.4. A straightforward way of achieving this is through *basic composition*, in which the sequential application of two differentially private mechanisms results in using up a total privacy budget of the sum of the ε values for each individual mechanism. If we have k ε-DP mechanisms that we run sequentially through our dataset, then the full process will be $k\varepsilon$-differentially private. More generally, for k DP mechanisms that each have potentially different ε values, the total process will be $\sum_i \varepsilon_i$-DP, for $1 \leq i \leq k$. There are actually multiple different composition theorems which can improve upon basic composition when using *approximate differential privacy* (e.g. Advanced Composition of Dwork et al. (2010), also see Section 2.2.4 on the Moments Accountant).

Finally, a third useful property of differential privacy is *group privacy*. This is the scenario in which the neighboring datasets D and D' differ, not in one entry, but in several. In this case, the privacy guarantee shows a linear drop in the number of entry differences. More formally, if a DP mechanism $\mathcal{M} : D^n \to Y$ is $k\varepsilon$-DP, and D, D' differ in k positions, then for all $S \subseteq \text{Range}(\mathcal{M})$:

$$\Pr[\mathcal{M}(D) \in S] \leq e^{k\varepsilon} \Pr[\mathcal{M}(D') \in S] \tag{2.7}$$

This can be useful if, for instance, a dataset contains multiple records that belong to a single individual.

Approximate differential privacy

So far, we have described what is known as *pure differential privacy* with Definition 2.1 and Equations 2.1 and 2.3. It turns out we can actually 'loosen' our privacy guarantees provided by pure differential privacy without really giving up privacy, but at the same time improving utility in our algorithms. This is especially the case for running many algorithms sequentially, as in the case of training a machine learning model for multiple epochs. For this *approximate differential privacy*, we take our original ε-DP guarantee and we add a 'cryptographically small' probability that it will not work. It turns out that this is enough to often significantly improve the utility for our DP mechanism.

To define approximate DP more formally, we take the original Definition 2.2.3 and slightly modify it with the extra term δ, originally introduced in Dwork et al. (2006a).

Definition 2.2.6. *For $\varepsilon \geq 0$ and $\delta \in [0, 1]$, a mechanism $\mathcal{M}$ is (ε, δ)-differentially private if, for all $S \subseteq Range(\mathcal{M})$, and for any two neighboring datasets D and D', the following holds true:*

$$\Pr[\mathcal{M}(D) \in S] \leq e^{\varepsilon} \Pr[\mathcal{M}(D') \in S] + \delta \tag{2.8}$$

In the case of $\delta = 0$, we go back to our original pure differential privacy definition.

To link this definition with the privacy loss random variable from Eqn. 2.2 and 2.3, we again keep the definition exactly the same, except that the random variable is bounded by ε with probability $1 - \delta$, as opposed to probability 1:

$$\max_{\forall y} \left| \mathcal{L}^{(y)}_{\mathcal{M}(D)||\mathcal{M}(D')} \right| \leq \varepsilon, \quad \text{w.p.} \quad 1 - \delta \tag{2.9}$$

Choosing δ

As with ε, the question arises of which value to actually set δ to. By definition, δ is a bad scenario, in which potentially anything could be released about the dataset. As discussed within the DP literature (e.g. Mironov (2017)), a good rule of thumb is to set $\delta \ll \frac{1}{n}$, with n being the number of individuals in the dataset D. To motivate this point, we can look at a hypothetical example of a mechanism that provides a guarantee of $(0, \delta)$-DP. In other words, this mechanism is perfectly private, except with a small probability δ.

Definition 2.2.7. *Let $\mathcal{M}$ be a randomized mechanism and D be a dataset consisting of n individuals. For each element in the dataset $x_i \in D$, output x_i with probability δ and do nothing with probability $1 - \delta$.*

For any particular individual in the dataset, with this mechanism $\mathcal{M}$ we are risking releasing their data point with probability δ. Since we run this n times, over every single individual in D, the total probability that we do not release any individual's data point is $(1 - \delta)^n$. Conversely, the probability that we *do* release somebody's data point is $1 - (1 - \delta)^n$. Under this hypothetical mechanism, we can see that if we set δ to be around $\frac{1}{n}$, then this probability is approximately 0.63. As we increase δ, this probability approaches 1; however, if we set δ to be 'much less than n' (e.g. $\delta = \frac{1}{n^2}$), then this probability of releasing someone's data point is nearly 0. While this hypothetical mechanism shows a worst-case scenario of the consequences of using a high δ, it can guide the data analyst in selecting a safe δ value.

ℓ_2-sensitivity

The question again arises of how to achieve a particular (ε, δ)-DP guarantee for a given query f. In contrast to the case of pure differential privacy, here we use a

slightly different definition for sensitivity, being the ℓ_2-sensitivity as opposed to the ℓ_1-sensitivity above.

Definition 2.2.8. *Given a function $f : D^n \to \mathbb{R}^k$ that takes in a dataset D and outputs a vector of dimension k, the ℓ_2-sensitivity of the function is as follows:*

$$\Delta_2^{(f)} = \max_{D,D'} ||f(D) - f(D')||_2, \tag{2.10}$$

for all neighboring datasets D, D'.

We are again looking at the maximum amount by which an algorithm f can change, when one data point is altered. In this case, however, we measure this distance between neighboring datasets with the ℓ_2 norm. In comparison with using the ℓ_1 norm, we can obtain a lower sensitivity in this case, which will in turn allow us to add less random noise to our query. For a given output of f, $y \in \mathbb{R}^k$, we have the following inequality: $||y||_2 \leq ||y||_1 \leq \sqrt{k}||y||_2$. This means that, in addition to the ℓ_2 norm being smaller than the ℓ_1 norm, the two are at most a factor of $\sqrt{k}$ apart.

Gaussian mechanism

The standard mechanism for achieving approximate differential privacy is the *Gaussian Mechanism*. The Gaussian Distribution is another continuous probability distribution with mean parameter μ and variance parameter σ^2, as shown in Figure 2.2. It is defined as follows:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{2.11}$$

We again center the distribution at $\mu = 0$.

In order to achieve a particular (ε, δ)-DP guarantee, we add noise from the Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ to each coordinate of our query f, calculated based on the ℓ_2-sensitivity of f. More formally, this can be seen in Definition 2.2.9.

Definition 2.2.9. *The Gaussian mechanism is defined for a given function $f : D^n \to \mathbb{R}^k$, $\varepsilon \in (0, 1)$ and $\delta \in (0, 1)$ as follows:*

$$\mathcal{M}_L(D, f(\cdot), \varepsilon) = f(D) + (Y_1, \ldots, Y_k), \tag{2.12}$$

where each random variable Y_i is drawn i.i.d. from $\mathcal{N}(0, 2\ln(\frac{1.25}{\delta})\frac{\Delta_2^2}{\varepsilon^2})$.

With this mechanism, we can achieve a desired (ε, δ)-DP guarantee for our query f.

Theorem 2.2.2. *The Gaussian mechanism preserves (ε, δ)-differential privacy.*

Proof. We refer to the proof of Theorem A.1 in Dwork and Roth (2013). □

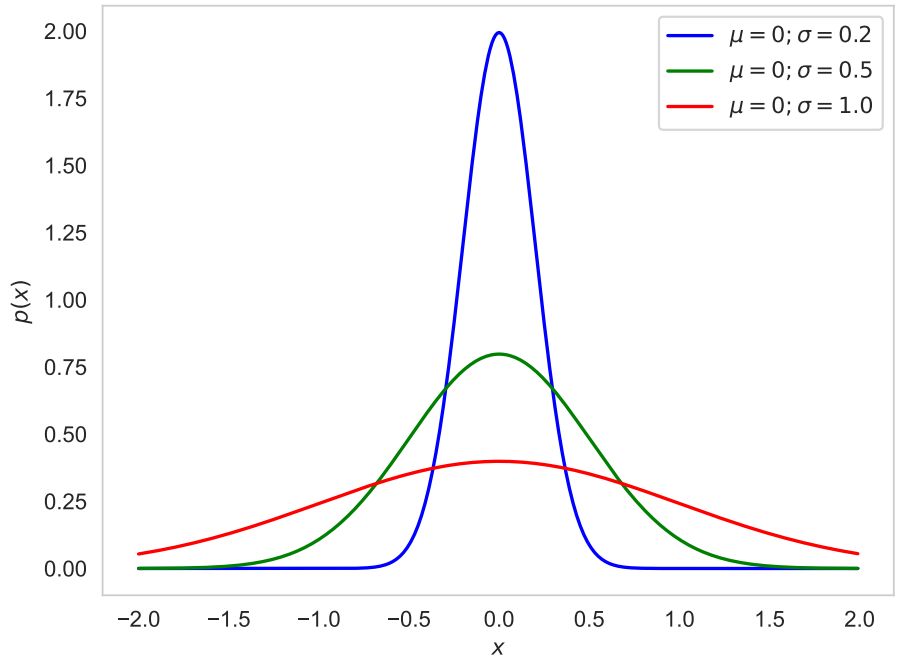


Figure 2.2: Three instantiations of the Gaussian distribution, all centered at 0 ($\mu = 0$), with varying standard deviation parameters σ. As we decrease σ, there is more probability mass around μ (e.g. blue curve at $\sigma = 0.2$). As σ is increased, the probability mass becomes more spread out, with lower peaks (e.g. red curve at $\sigma = 1.0$). In comparison to the Laplace distribution, the peak of the Gaussian distribution is far less sharp, meaning that the probability mass is more spread out away from the mean. The tails of the Gaussian are also lighter compared to Laplace, with more extreme values expected to appear for the latter. As with the Laplace distribution, the Gaussian is also symmetrical around its mean and unimodal.

Exponential mechanism

While not utilized for the investigations in this thesis, an additional popular mechanism in differential privacy is the *exponential mechanism* (McSherry and Talwar, 2007), used for selecting the "best" element from a given set $\mathcal{R}$ in a differentially private manner. This is achieved by defining a *score function* q, which assigns a score to a dataset D and each element $r \in \mathcal{R}$: $q(D, r)$. In order to provide a DP guarantee, the set element r that is returned by the mechanism is *not always* the one with the highest score, but rather with probability proportional to $\exp(\frac{\varepsilon q(D,r)}{2\Delta^{(q)}})$. In other words, higher scoring outputs are exponentially more likely, depending on the privacy budget ε and the sensitivity $\Delta^{(q)}$ of q.

The exponential mechanism is in fact a fundamental mechanism in differential privacy, with which all other mechanisms can be defined, including the Laplace and Gaussian mechanisms above. For example, in the case of the Laplace mechanism applied to a query that has a single output value, the set $\mathcal{R}$ is simply the set of real

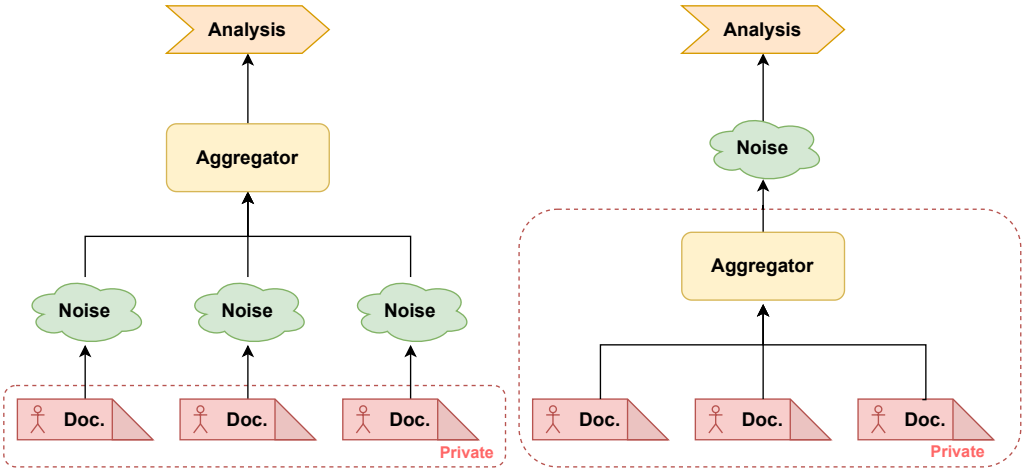


Figure 2.3: Local DP (left) vs. global DP (right). In the local framework, the aggregator does not have access to the original data, with each individual applying DP to their own private data point. In the global framework, the aggregator adds DP noise to the original data, given a specific query from an analyst.

numbers $\mathbb{R}$.

Global DP vs. Local DP

One final important point to address with respect to differential privacy is the difference between **global**, also known as **centralized** differential privacy, and **local** differential privacy. The distinction between the two is depicted in Figure 2.3. To explain this distinction, we first define the notion of a *trusted aggregator*, also known as a *trusted curator*.

Definition 2.2.10. *A **trusted aggregator** is a trusted party that aggregates the data from the data generators (i.e. the individuals providing the data) in a dataset D. They can answer any given query about the dataset, providing the differentially private answer to it.*

In the case of global differential privacy, the query $f(D)$ is first evaluated, and then perturbed by the trusted data aggregator. This means that the trusted aggregator is able to observe the private information of all individuals, and is trusted to correctly apply the DP mechanism to the query on the dataset. There are many scenarios in which such a party would indeed be considered trustworthy. For instance, if a hospital is providing medical data about its patients for statistical analysis, it has legal requirements to protect the privacy of those patients, as described in Section 1.1. The hospital thus acts as the trusted curator and it is in its best interests not to have any privacy breaches of the provided data.

In contrast, in **local differential privacy (LDP)** *each individual data holder perturbs their own data point*, prior to data collection and without relying on a third party. In this case, the aggregator of the data receives already noisified data points from individual data holders and can then pass them on for further statistical

analysis. A classic example of local differential privacy is the **randomized response** mechanism, outlined below in Section 2.2.3. Importantly, as noted by Wang et al. (2020a), in LDP *any two data points* are considered neighboring, in contrast to the global DP definition of two datasets differing in one record, as in Definition 2.2.1.

Since each individual is fully in control of the privatization process, this makes LDP a particularly attractive setting for providing a privacy guarantee. The difficulty, however, is that we typically require orders of magnitude more perturbation for our query f than we otherwise would need in the global DP setting, since the notion of adjacency between two data points is far stricter in LDP. We refer to this as the *strict adjacency problem* and discuss this in detail in Chapter 6.

With regards to the structure of this thesis, Chapters 3 and 4 address NLP from the perspective of global DP, while Chapters 5 and 6 investigate local DP. More specifically, **RQ1** and **RQ2**, dealing with the privatization of NLP models, fit into the global DP scenario, utilizing the global DP algorithm *differentially private stochastic gradient descent (DP-SGD)*, described in Section 2.2.4. The individual data holder provides their original data in the form of a textual document to the machine learning model, with DP perturbation occurring during the model training process. In contrast, **RQ3** deals with the privatization of the text itself. In this case, a given text is rewritten locally by a data holder using a text rewriting model, with a certain (ε, δ)-differentially private guarantee. After the data point has been privatized, it can then be used for any downstream analysis, due to the post-processing property of DP, outlined above.

2.2.2 Example application of DP in the global setting

To motivate the above concepts of differential privacy in a more concrete manner, we provide a specific pedagogical example of a simple DP mechanism applied to a hypothetical dataset. While this example is simpler than the machine learning and NLP setting described in Sections 2.2.4 and 2.2.5, it illustrates the fundamental building blocks of DP that will be applied to NLP models and textual datasets throughout this thesis.

Let us define an example dataset X as follows, shown in Table 2.1. Each row

Name	**Trait A**	...	**Trait K**
Alice	0	...	1
Bob	0	...	1
Claire	1	...	0
⋮	⋮	⋱	⋮
Zane	1	...	0

Table 2.1: Sample dataset X, consisting of n individuals, each having k sensitive attributes that we want to protect. Each attribute is a binary value of either 0 or 1, representing the presence or absence of that attribute for a given individual.

of this dataset is associated with a specific individual, with n individuals and k

attributes in total. The first column shows the name of a particular individual, with subsequent columns consisting of binary values $X_{i,j} \in \mathbb{Z}_2$, representing the presence or absence of a particular sensitive trait j associated with individual i. For instance, the first attribute 'Trait A' could represent whether the individual has a certain medical condition or not.

The first *query* we may want to answer on this dataset is: How many people in our dataset possess trait A? This is a simple sum query, in which we add up all of the binary values of the *Trait A* column. More formally, let $g(X) = \sum_{i=1}^{k} X_i$ be a function that sums up a set of bits X_i, for each individual i.

First, we ask what is the ℓ_1-sensitivity of this query. In other words, if one individual is removed from this dataset, what is the maximum value by which the sum query can change? Intuitively, we can see that this is simply 1. Either we have removed an individual with a value of 1 for the *Trait A* column, in which case the output from our neighboring dataset will be different by 1, or we have removed an individual with a value of 0, resulting in the same output for g. Hence, we can see that $\Delta_1^{(g)} = 1$.

Next, we can use the Laplace mechanism from Eqn. 2.6 to calculate the amount of noise we need to add to the true answer of our sum query g. We thus must add noise drawn from $\text{Lap}(\frac{1}{\varepsilon})$ to our true answer $g(X)$. Our randomized mechanism $\mathcal{M}$ is therefore: $\mathcal{M}(X) = g(X) + \text{Lap}(\frac{1}{\varepsilon})$.

Let us assume that we have a fairly large dataset of 5000 individuals. If the true answer to this query for our dataset is 1217, then the differentially private answer at $\varepsilon = 1$ could be, for instance, something around 1219. This provided DP guarantee can be seen more concretely in Figure 2.4.

With our DP mechanism, the possible outputs for the original dataset, and those for a neighboring dataset, are shown as the blue and green probability density function, respectively. We can see that the ratio of these two, at any point $\mathcal{M}(D)$, is always bounded by e^{ε}. If we increase ε, then this ratio also increases, with sharper peaks around the true values. Similarly, decreasing ε results in the two distributions being more similar, with the probability density functions being more stretched out.

We can look at this from the perspective of an adversary that is trying to figure out which of the two databases is the real one, X or X', given an output. When the blue curve is higher than the green curve (e.g. at $\mathcal{M}(D) = 1218$), the adversary may suspect that the true dataset is X, since it has a higher probability that the observed output came from it. **Specifically, the adversary is at most $\mathbf{e}^{\varepsilon}$ times more certain that the output came from the true database, as opposed to from the neighboring one.**

Conversely, when the green curve is higher than the blue curve, i.e. when $\mathcal{M}(X') > \mathcal{M}(X)$, then the adversary is actually deceived that the true dataset is X'! In this case, the probability of an output **coming from the neighboring dataset** X' is at most e^{ε} times more likely than from the dataset X.

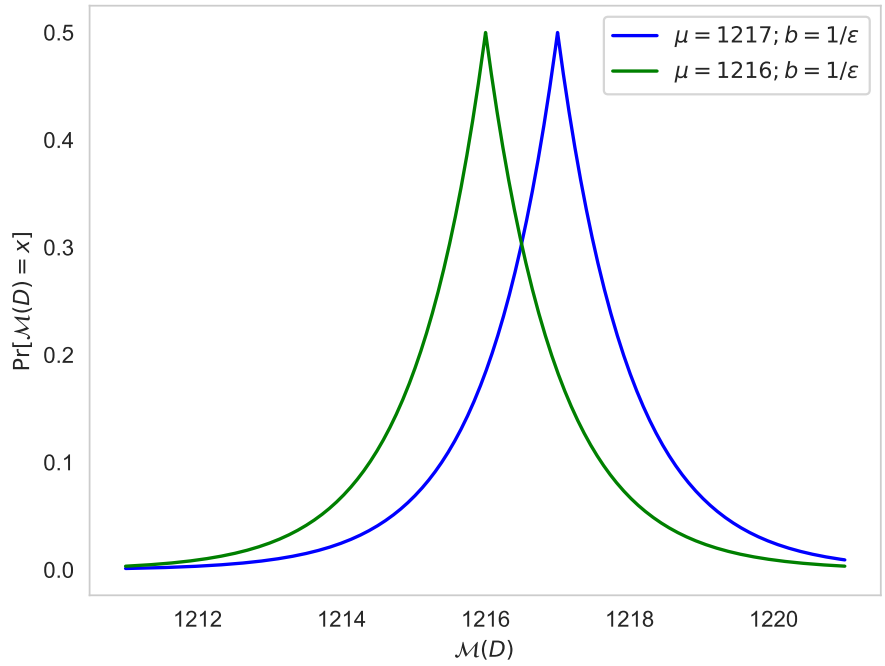


Figure 2.4: The possible outputs of our DP mechanism $\mathcal{M}$ for our target dataset X (blue curve) and a neighboring dataset X' (green curve). The x axis shows the possible output values of the mechanism, given an arbitrary dataset D, while the y axis shows the associated probability density for a given output value x. The ratio of the blue curve to the green curve at any given point never exceeds e^{ε}, and never goes below $e^{-\varepsilon}$. Since $\varepsilon = 1.0$ in this case, this ratio is always bounded by e (or $\frac{1}{e}$ when $\mathcal{M}(X') > \mathcal{M}(X)$).

We can additionally view this from the perspective of the privacy loss random variable from Eqn. 2.2 in Figure 2.5, shown as the cumulative distribution function of $\mathcal{L}_{\mathcal{M}(X)||\mathcal{M}(X')}$. We can obtain this plot from Figure 2.4 above by taking the ratio of probabilities for the blue and green curves, $\mathcal{M}(X)$ and $\mathcal{M}(X')$, at all possible points, and ordering them. Here we can very concretely see the privacy guarantee that DP provides, where the privacy loss is clearly bounded between $-\varepsilon$ and ε for our mechanism, in this case setting $\varepsilon = 1$. The corresponding probability density function of $\mathcal{L}_{\mathcal{M}(X)||\mathcal{M}(X')}$ would similarly show these bounds of $-\varepsilon$ and ε, with two peaks at each of these two values.

Going back to our sample dataset from Table 2.1, let us expand our original sum query g to include more attributes. Concretely, we want to ask the question: "How many people in our dataset possess *each* trait?". In this case, we are asking for the sum of each column, hence our output will be a vector of dimension k, with each element representing the sum of binary values for a particular column. Let $h(X) = (g(X_{1,*}), g(X_{2,*}), \ldots, g(X_{k,*}))$, where $X_{j,*}$ represents the values of all individuals for trait j.

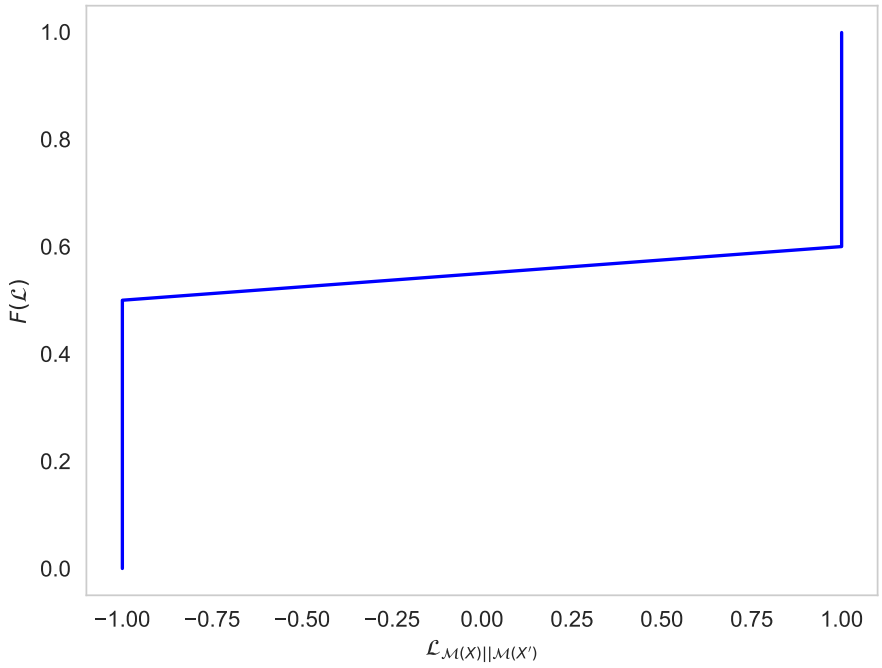


Figure 2.5: The cumulative distribution function $F(\mathcal{L})$ of the privacy loss random variable $\mathcal{L}_{\mathcal{M}(X)||\mathcal{M}(X')}$ at $\varepsilon = 1.0$. We can see that there is a lot of probability mass at the points $-\varepsilon$ and ε, with some transition in between. This clearly demonstrates that $\mathcal{L}_{\mathcal{M}(X)||\mathcal{M}(X')}$ never goes outside of these bounds, meaning that the contribution of any single individual to the analysis, in this case our original sum query $g(X)$ with added random noise drawn from $\text{Lap}(\frac{1}{\varepsilon})$, is strictly bounded.

Here the ℓ_1-sensitivity of h depends on the binary values of multiple columns. Intuitively, in the worst-case scenario we can think of having removed an individual with values of 1 for all columns and replaced them with another individual with values of 0 for all columns. More formally, recall from Eqn. 2.4 that $\Delta_1^{(h)} = \max_{X,X'} ||h(X) - h(X')||_1$, for two neighboring datasets, X and X'. Then:

$$\begin{aligned}
\Delta_1^{(h)} &= \max_{X,X'} ||h(X) - h(X')||_1 \\
&= |g(X_{1,*}) - g(X'_{1,*})| + \cdots + |g(X_{k,*}) - g(X'_{k,*})| \\
&\leq ||\mathbf{1}||_1 \\
&= \underbrace{1 + \cdots + 1}_{k \text{ times}} \\
&= k
\end{aligned} \tag{2.13}$$

The ℓ_1-sensitivity is thus simply the number of output dimensions of our query, in this case the number of traits.

We can again use the Laplace mechanism to calculate the amount of noise we need to add to the true answer of our query h. This time, however, we will

need to add multiple Laplace random variables to our output, since the output of h is a multi-dimensional vector of size k. Our privatized answer would then be $h(X) + (Y_1, \ldots, Y_k)$, with each Y_i drawn i.i.d. from $\text{Lap}(\frac{k}{\varepsilon})$, based on our obtained ℓ_1-sensitivity in Eqn. 2.13 above.

For comparison, we can try using the Gaussian mechanism for privatizing h instead. Recall that the Gaussian mechanism calculates the scale of the noise to be added to the true answer of a query based on the ℓ_2-sensitivity (Def. 2.2.9). The ℓ_2-sensitivity of h is calculated as follows:

$$\begin{aligned}
\Delta_2^{(h)} &= \max_{X,X'} ||h(X) - h(X')||_2 \\
&= \sqrt{(g(X_{1,*}) - g(X'_{1,*}))^2 + \cdots + (g(X_{k,*}) - g(X'_{k,*}))^2} \\
&\leq ||\mathbf{1}||_2 \\
&= \underbrace{\sqrt{1 + \cdots + 1}}_{k \text{ times}} \\
&= \sqrt{k}
\end{aligned} \tag{2.14}$$

It turns out here that, with the Gaussian mechanism, we can add a lot less noise to our query h than with the Laplace mechanism, depending on the exact size of k. In fact, as we increase the number of output dimensions k, we can expect a better utility improvement when using the Gaussian mechanism due to the greater difference in the added noise scale. More concretely:

$$\text{Laplace: } \tilde{y} = h(X) + \text{Lap}(\tfrac{k}{\varepsilon})$$

$$\text{Gaussian } \tilde{y} \approx h(X) + \mathcal{N}(0, (\tfrac{\sqrt{k}}{\varepsilon})^2)$$

Of course, we are sacrificing some privacy guarantees by including the extra δ probability. The exact choice of which of these mechanisms to use can be task-specific, and either could be a better option depending on the exact scenario.

Our privatized answer to our multi-dimensional sum query h is thus $h(X) + (Y_1, \ldots, Y_k)$, with each Y_i drawn i.i.d. from $\mathcal{N}(0, 2\ln(\frac{1.25}{\delta})(\frac{\sqrt{k}}{\varepsilon})^2)$. Recall from Section 2.2.1 that it is best to set $\delta \ll \frac{1}{n}$, where n is the number of individuals in the dataset. Again, assuming our dataset from Table 2.1 has 5000 individuals, we can thus set δ to be far less than $\frac{1}{5000}$, e.g. $\delta = 10^{-5}$. Setting $\varepsilon = 0.999$ and with $k = 200$, the noise for each Y_i would be drawn from approximately $\mathcal{N}(0, (166.0)^2)$. In contrast, our Laplace scale would be $b = \frac{k}{\varepsilon} \approx 200$.

We can additionally see that, if we increase the size of our dataset, the added noise to the true answer of our query will have less impact on our final output. If our dataset contains 50000 individuals, then since we still add the same amount of noise to the output of our query as with a smaller dataset, we will have a more accurate final answer. Conversely, with a smaller dataset of 500 individuals, our final noisy answer will be heavily skewed from the true answer.

We have thus shown an example application of differential privacy to a simple dataset, both in the pure and approximate DP settings. We saw how our privacy guarantee can be interpreted from the perspective of the *privacy loss random variable*, and discussed some comparisons between the Laplace and Gaussian mechanisms for achieving our differential privacy guarantee. The application above has been an example of *global* DP, in which we add noise to the output of our query. As *randomized response* is another good pedagogical example of DP, this time for the *local* DP setting, and comes up frequently in discussions on differential privacy, we present it in the section below.

2.2.3 Randomized Response

The randomized response technique can be seen as the *oldest* differentially private algorithm, having been proposed by Warner (1965), with further modifications by Greenberg et al. (1969). While there are different variations on the exact formulation of the method, we will follow a generalized form as presented in Dwork and Roth (2013).

We can imagine a setting as follows. A professor in charge of a large class has administered an exam. He suspects that a certain number of students in the class have cheated on the exam. He would like to make an estimate of the number of students that cheated; however, he cannot ask the students directly, since an affirmative answer would potentially carry negative consequences in this setting. In the non-private scenario, such a survey would therefore be expected to collect mostly negative answers. What can be done to obtain an accurate estimate to the desired query, without individuals giving up their sensitive information?

We can formulate this problem more formally as follows. There are n students, each student i with a sensitive 'bit' of information $X_i \in \{0, 1\}$ that represents whether or not they cheated on the exam. If the students provide their bit X_i directly, this would be a privacy violation. Therefore, each student generates a value Y_i, which depends on the true value X_i, but with some randomness incorporated into it. This Y_i is then the answer that is sent off to the data analyst, in this case the professor in charge of the class. The goal of the analyst is to calculate the percentage of students that cheated, formally:

$$f = \frac{1}{n} \sum_{i=1}^{n} X_i \tag{2.15}$$

The way to incorporate randomness into X_i, in order to obtain Y_i, is as follows. Each student in the class gets a coin. They flip this coin in secret, so that only they know the result of the flip. If it lands heads, then they provide the true answer, setting $Y_i = X_i$. If it lands tails, then they flip the coin again in secret. Based on the outcome of this second coin flip, they then will say 'yes' or 'no' to the question of whether they cheated or not, for the flip outcome of heads or tails, respectively.

More formally, we define a parameter $p \in (0, \frac{1}{2})$. We determine Y_i as follows:

$$Y_i = \begin{cases} X_i, & \text{w.p.} \quad \frac{1}{2} + p \\ 1 - X_i, & \text{w.p.} \quad \frac{1}{2} - p \end{cases} \tag{2.16}$$

We can see that, as we approach $p = \frac{1}{2}$, then we go back to the original non-private scenario, where each individual simply gives up their private bit X_i to the analyst, setting $Y_i = X_i$. This fully accurate answer would provide the data analyst with the best utility for the original query, but at the cost of privacy for the data providers. In contrast, we can see that if we approach $p = 0$, then we move towards perfect privacy, but no utility, since the bit Y_i is fully random, independent of X_i. Both outcomes of $Y_i = 0$ and $Y_i = 1$ can thus occur with uniform probability.

To find a middle-ground, we can set $p = \frac{1}{4}$. We would therefore expect a truthful answer $Y_i = X_i$ with probability $\frac{3}{4}$. While this is certainly not as private as setting $p = 0$, we are able to retain utility of the query to the data analyst. This again reflects the typical problem within differential privacy research of finding a good balance between the privacy and utility of an algorithm. As we set p closer to 0, we approach maximum privacy and minimum utility. In contrast, as $p \to \frac{1}{2}$, we approach maximum utility and minimum privacy. We then calculate the noisy answer to our query, $\tilde{f}$, as follows: $\tilde{f} = \frac{1}{n} \sum_{i=1}^{n} Y_i$.

We can calculate the estimate that the analyst will obtain of the true proportion of individuals that cheated f, from the aggregated Y_i values as follows. Since we are obtaining a biased estimate of the true answer to the query f by incorporating the above noise mechanism from Eqn. 2.16, we need to adjust this in the final answer. We want the calibrated correct proportion of students that cheated, defined as f', where $\mathbb{E}[f'] = f$. This means that we want an expression for f' in which we get an unbiased estimate for f. First, note that the expected value of our noisy bits Y_i can be calculated as follows.

$$\begin{aligned} \mathbb{E}[Y_i] &= X_i \left(\frac{1}{2} + p\right) + (1 - X_i) \left(\frac{1}{2} - p\right) \\ &= 2pX_i + \frac{1}{2} - p \end{aligned} \tag{2.17}$$

We can rearrange this expression to calculate a given X_i, due to linearity of expectations:

$$X_i = \mathbb{E}\left[\frac{1}{2p}\left(Y_i - \frac{1}{2} + p\right)\right] \tag{2.18}$$

Plugging this into the original formula for f in Eqn. 2.15, we obtain an unbiased estimator as follows:

$$f' = \frac{1}{n} \sum_{i=1}^{n} \left[\frac{1}{2p}\left(Y_i - \frac{1}{2} + p\right)\right] \tag{2.19}$$

This will give us the estimate of the correct proportion of students that cheated, $\mathbb{E}[f'] = f$, since taking the expectation of f' in Eqn. 2.19 leads to each individual summand equivalent to X_i, as in Eqn. 2.18, due to linearity of expectations.

Importantly, randomized response provides each individual with **plausible deniability**. This means that the individual is able to credibly claim that they did not make a certain statement. In the above scenario, if a student answered 'yes' to the question of whether they cheated or not on the exam, they can respond that their answer is simply due to the outcome of the second coin flip. As we decrease the p parameter, individuals approach maximum deniability, and vice versa. This setting can in fact be translated to finding out any sensitive information about a group of people (e.g. taboo behavior that people would not want to disclose), while maintaining each individual's confidentiality.

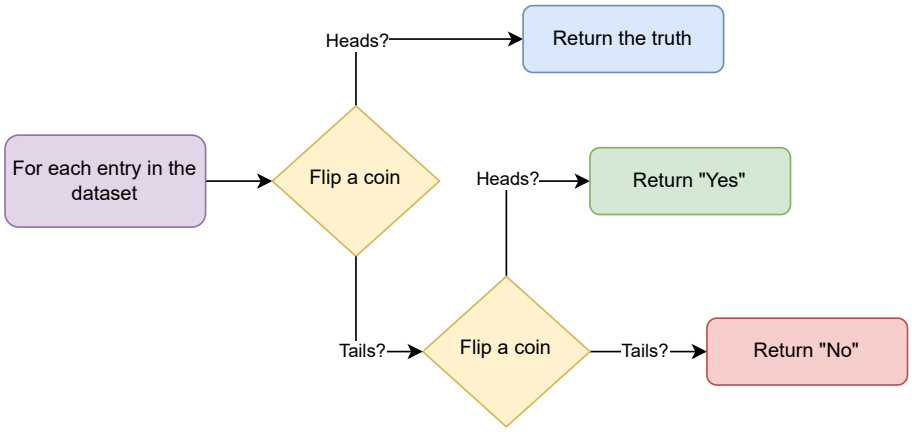


Figure 2.6: The randomized response procedure. Each individual in the dataset flips a coin. If it lands *heads*, then they tell the truth. If it lands *tails*, then they flip another coin. They then reply "Yes" or "No" to the analyst's question, based on the outcome of this second coin.[1]

The overall randomized response procedure is outlined in Figure 2.6. We can in fact translate this to the language of differential privacy to get a concrete ε-DP guarantee, for a given value of p. Following the notation from Eqn. 2.1, we define our dataset of individuals $D \in \{0,1\}^n$ as the string of bits $X_1, \ldots, X_n$, with each $X_i \in \{0,1\}$. We then define our mechanism $\mathcal{M}$ as randomized response, according to Eqn. 2.16. Let $y \in \{0,1\}^n$ be the resulting string of bits $Y_1, \ldots, Y_n$. Then for two neighboring datasets D and D', i.e. where both differ in a single coordinate j, the privacy loss random variable (Eqn. 2.2) will be as follows:

$$\begin{aligned} \mathcal{L}^{(y)}_{\mathcal{M}(D)||\mathcal{M}(D')} &= \ln\left(\frac{\Pr[\mathcal{M}(D)=y]}{\Pr[\mathcal{M}(D')=y]}\right) \\ &= \ln\left(\frac{\prod_{i=1}^{n}\Pr[Y_i=y_i]}{\prod_{i=1}^{n}\Pr[Y'_i=y_i]}\right) \\ &= \ln\left(\frac{\Pr[Y_j=y_j]}{\Pr[Y'_j=y_j]}\right) \end{aligned} \tag{2.20}$$

We obtain the last equality due to the fact that only one coordinate is different in the products of the numerator and denominator, with all other terms being common

[1] Figure based on https://towardsdatascience.com/understanding-differential-privacy-85ce191e198a.

and thus cancelling out. Finally, from Eqn. 2.16, we can see that for any possible outcomes of Y_j, the final ratio is bounded as follows:

$$\mathcal{L}^{(y)}_{\mathcal{M}(D)||\mathcal{M}(D')} = \ln\left(\frac{\Pr[Y_j = y_j]}{\Pr[Y'_j = y_j]}\right) \leq \frac{\frac{1}{2}+p}{\frac{1}{2}-p} \tag{2.21}$$

From the definition of ε with regards to the privacy loss random variable in Eqn. 2.3, we can therefore see that the above bound in Eqn. 2.21 is equivalent to the privacy budget, $\varepsilon = \frac{\frac{1}{2}+p}{\frac{1}{2}-p}$. As a concrete example, if we set $p = \frac{1}{4}$, thus using fair coin flips, our randomized response mechanism is $(\varepsilon = \ln 3)$-differentially private.

We have thus demonstrated a concrete example of a differentially private mechanism in the local setting. In contrast to the global setting, here we do not need to rely on a trusted aggregator to apply the differentially private mechanism, with each individual (e.g. student in the above exam setting) introducing randomness into their own data point. We will see this same setting of LDP, specifically applied to textual data used for NLP models in Chapters 5 and 6. While the data is more complex, the underlying principle remains the same: Any individual can take their own data point and perturb it to achieve a given ε-DP guarantee. This data point can then be used in any downstream analysis without violating that individual's privacy beyond the provided DP guarantee due to the property of DP being closed under post-processing.

2.2.4 DP and Machine Learning

Having gone over differential privacy in the general statistics setting, we now turn to machine learning. There have been various applications of DP in the machine learning and deep learning settings (Chaudhuri et al., 2011; Rubinstein et al., 2012; Shokri and Shmatikov, 2015; Papernot et al., 2017; McMahan et al., 2018; Wang et al., 2021), but the most relevant for our purposes is the algorithm *differentially private stochastic gradient descent (DP-SGD)* by Abadi et al. (2016b), discussed in detail below. First, however, we present a general discussion of how one may go about applying differential privacy to a neural model.

As outlined in Figure 2.7, we have multiple options available. First, we may consider privatizing the data itself, also known as *input perturbation*. This would be an instance of local differential privacy, in which we remove identifiable information from an input document before it is even sent off to the network. For example, this could be an input textual document which we rewrite with differential privacy guarantees. This methodology will in fact make up the core of Chapters 5 and 6, where another neural model with noise added to its hidden representations is used to rewrite a given document.

Next, we may consider adding noise to the model parameters, or some hidden layer of the model. As mentioned above, we will utilize the latter approach in order to obtain privatized data for our text rewriting methods.

In addition, it is also possible to privatize the output of a model, known as

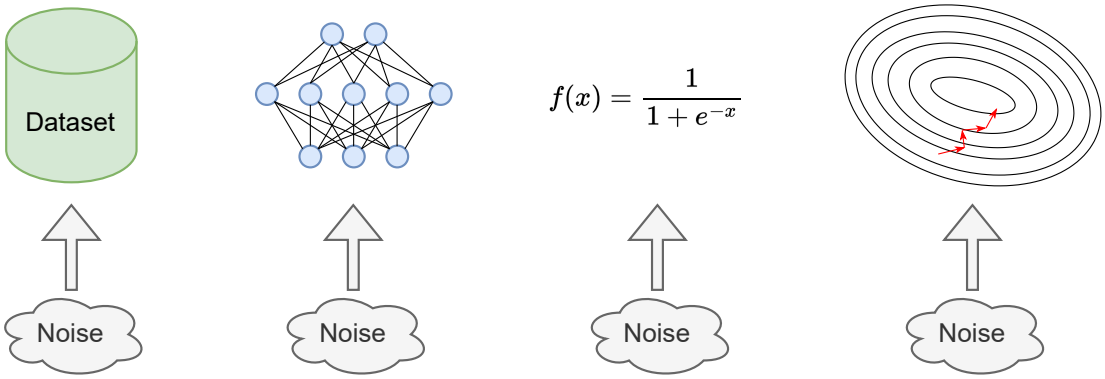


Figure 2.7: Different options for applying differential privacy to the machine learning process. Random noise can be added directly to the dataset itself (far left), to model parameters or a hidden layer of the model (second from the left), the output of the model (second from the right), or the optimization process when training the model (far right).[2]

output perturbation (Chaudhuri et al., 2011; Rubinstein et al., 2012). In this case, it is important to be careful about the exact setting in which such a model is deployed, as it cannot be released in a white-box manner. One reason for this is that the parameters of the model are vulnerable to recovery of the input data that was originally used to train it, for instance using membership inference, or model inversion-type attacks (Fredrikson et al., 2014, 2015; Hitaj et al., 2017; Shokri et al., 2017).

Finally, we may consider adding noise to the optimization process. This includes *objective perturbation*, in which we add noise to the loss function that is being optimized (Chaudhuri et al., 2011; Kifer et al., 2012), as well as *gradient perturbation*, in which we add noise to the gradient of the loss function. This latter method is in fact the approach of DP-SGD, and comprises the main methodology used in Chapters 3 and 4. As we are training a model, instead of taking a step down the true gradient of the loss function, we take a noisy step that obscures the contribution of the original training data.

DP-SGD

The notion of adding noise to the gradient of a loss function in order to achieve a differentially private guarantee was originally proposed by Williams and McSherry (2010) for logistic regression, with further developments by Jain et al. (2012); Song et al. (2013). It was then developed in detail in terms of its theoretical properties by Bassily et al. (2014) and then extended to the deep learning setting in Abadi et al. (2016b) as *differentially private stochastic gradient descent (DP-SGD)*.

When using DP-SGD, two additional steps are introduced to the standard stochas-

[2] Figure based on https://medium.com/bluecore-engineering/differential-privacy-in-the-real-world-f31a5df1398f.

tic gradient descent algorithm (Robbins and Monro, 1951). For a given input x_i from the dataset, we obtain the gradient of the loss function $\mathcal{L}(\theta)$ at training time step t, $g_t(x_i) = \nabla_{\theta_t}\mathcal{L}(\theta_t, x_i)$. We then clip this gradient by ℓ_2 norm with clipping threshold C in order to constrain its range (i.e. ℓ_2-sensitivity, Eqn. 2.10), limiting the amount of noise required for providing a differential privacy guarantee.

$$\bar{g}_t(x_i) = \frac{g_t(x_i)}{\max\left(1, \frac{||g_t(x_i)||_2}{C}\right)} \tag{2.22}$$

Subsequently, we utilize the Gaussian mechanism (Eqn. 2.12), adding Gaussian noise to the gradient to make the algorithm differentially private. This DP calculation is grouped into 'lots' of size L.

$$\tilde{g}_t = \frac{1}{L}(\sum_{i \in L} \bar{g}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I})) \tag{2.23}$$

The descent step is then performed using this noisy gradient, updating the network's parameters θ, with learning rate γ.

$$\theta_{t+1} = \theta_t - \gamma \tilde{g}_t \tag{2.24}$$

While originally defined for the SGD optimizer, this application of DP to the optimization process can be extended to other optimizers, such as Adam (Kingma and Ba, 2014). As Adam shares the core principle of gradient computing within SGD, to make it differentially private we can add noise to the gradient following Eqn. 2.23, prior to Adam's moment estimates and parameter update. Despite the conceptual simplicity of DP-SGD and extensions such as DP-Adam, the utilized privacy budget can quickly become very large under techniques such as basic composition, when training a neural model over many epochs. To address this, Abadi et al. (2016b) proposed the *moments accountant* as an advanced DP composition technique, which we present in detail below. Due to properties of the Gaussian mechanism allowing for advanced composition, it is thus possible to significantly reduce the expended privacy budget, without requiring to add considerably more noise to the gradient.

Moments Accountant

DP-SGD introduced two features, namely (1) a reverse computation of the privacy budget, and (2) tighter bounds on the composition of multiple queries. First, a common DP methodology is to pre-determine the privacy budget (ε, δ) and add random noise according to these parameters. In contrast, DP-SGD does the opposite: Given a pre-defined amount of noise (hyperparameter of the algorithm), the privacy budget (ε, δ) is computed retrospectively. Second, generally in DP, with multiple executions of a 'query' (i.e. a single gradient computation in SGD), we can simply sum up the ε, δ values associated with each query, such that for k queries with privacy budget (ε, δ), the overall algorithm is $(k\varepsilon, k\delta)$-DP. However, this naive composition leads to a very large privacy budget as it assumes that each query used up the maximum given privacy budget.

The simplest bound on a continuous random variable Z, the Markov inequality, takes into account the expectation $\mathbb{E}[Z]$, such that for $\varepsilon \in \mathbb{R}^+$:

$$\Pr[Z \geq \varepsilon] \leq \frac{\mathbb{E}[Z]}{\varepsilon} \tag{2.25}$$

To simplify notation, we set the privacy loss from Eqn. 2.9, $Z = \mathcal{L}^{(y)}_{\mathcal{M}(D)||\mathcal{M}(D')}$. Using the Chernoff bound, a variant of the Markov inequality, on the privacy loss Z, we obtain the following formulation by multiplying Eqn. 2.25 by $\lambda \in \mathbb{R}$ and exponentiating:

$$\Pr[\exp(\lambda Z) \geq \exp(\lambda\varepsilon)] \leq \frac{\mathbb{E}[\exp(\lambda Z)]}{\exp(\lambda\varepsilon)} \tag{2.26}$$

where $\mathbb{E}[\exp(\lambda Z)]$ is also known as the moment generating function.

The overall privacy loss Z is composed of a sequence of consecutive randomized algorithms $X_1, \ldots, X_k$. Since all X_i are independent, the numerator in Eqn. 2.26 becomes a product of all $\mathbb{E}[\exp(\lambda X_i)]$. Converting to logarithmic form and simplifying, we obtain:

$$\Pr[Z \geq \varepsilon] \leq \exp\left(\sum_i \ln \mathbb{E}[\exp(\lambda X_i)] - \lambda\varepsilon\right) \tag{2.27}$$

Note the moment generating function inside the logarithmic expression. Since the above bound is valid for any moment of the privacy loss random variable, we can go through several moments and find the one that gives us the lowest bound.

Since the left-hand side of Eqn. 2.27 is by definition the δ value, the overall mechanism is (ε, δ)-DP for $\delta = \exp(\sum_i \ln \mathbb{E}[\exp(\lambda X_i)] - \lambda\varepsilon)$. The corresponding ε value can be found by modifying Eqn. 2.27:

$$\varepsilon = \frac{\sum_i \ln \mathbb{E}[\exp(\lambda X_i)] - \ln \delta}{\lambda} \tag{2.28}$$

The overall DP-SGD algorithm, given the right noise scale σ and a clipping threshold C, is thus shown to be $(O(q\varepsilon\sqrt{T}), \delta)$-differentially private using this accounting method, with q representing the ratio $\frac{L}{n}$ between the lot size L and dataset size n, and T being the total number of training steps. We refer to Abadi et al. (2016b) for further details.

Overall, the moments accountant is closely related to Rényi Differential Privacy (RDP) (Mironov, 2017), which combines the notions of ε-DP, (ε, δ)-DP and concentrated differential privacy (Dwork and Rothblum, 2016; Bun and Steinke, 2016). The last is a relaxation of DP that allows for a tighter analysis of the Gaussian mechanism.

2.2.5 Differential Privacy and NLP

To conclude this section on differential privacy, we discuss the general methods that have so far been used in applying differential privacy to the domain of NLP. We

discuss relevant work for each part of the thesis in more detail in Chapters 3-6. For more background on general methodologies in NLP, we refer to Section 2.3.

The study of differential privacy in NLP is part of the more general study of privacy-preserving methods for textual data. These include methods of anonymization through adversarial learning (Li et al., 2018; Elazar and Goldberg, 2018; Coavoux et al., 2018; Friedrich et al., 2019), as well as other privacy frameworks such as federated learning (McMahan et al., 2017) (e.g. Ge et al. (2020); Sui et al. (2020); Zhang et al. (2022)), homomorphic encryption (Gentry, 2009) (e.g. Kim et al. (2022); Chen et al. (2022); Mihara et al. (2020); Podschwadt and Takabi (2020)), and secure multiparty computation (Cramer et al., 2015) (e.g. Reich et al. (2019); Resende et al. (2021); Adams et al. (2021)).

With regards to differentially private NLP, the primary research directions can be split up in various ways. This includes local DP vs. global DP methods, gradient-based privacy with algorithms such as DP-SGD vs. privacy of internal model representations (e.g. hidden layers, word vectors), and so forth. These various categories of research directions are not mutually exclusive and can be combined to form a given methodology (e.g. various studies looking into local DP with privacy of internal model representations, such as Weggenmann and Kerschbaum (2018)).

Here we will categorize work into four primary research trends:

1. **Differentially private language models**
2. **Differentially private model representations**
3. **Synthetic data generation with differential privacy**
4. **General discussion and analysis**

We will go through each of these in turn.

First, there has been an extensive amount of work on **differentially private language models**. This can further be subdivided into two primary groups: (a) pre-training of language models with DP and (b) fine-tuning of language models with DP. The former category has seen an increasing amount of research appear (Anil et al., 2022; Yin and Habernal, 2022; Hoory et al., 2021; Ponomareva et al., 2022; Wu et al., 2022; Shi et al., 2022a). Apart from drops in utility in the DP setting, one large challenge lies in the added computational burden of training a model with differential privacy. We discuss this in more detail in Section 4.5.6 of Chapter 4. In addition to studies looking into pre-training transformers with DP (Anil et al., 2022; Yin and Habernal, 2022; Hoory et al., 2021; Ponomareva et al., 2022; Wu et al., 2022), some works investigate other model types, such as RNN-based language models (Shi et al., 2022a). To improve utility, some studies investigate methods to relax the stricter requirements of the DP setting, including adjusting the amount of noise added to the gradient, depending on what tokens are present in an input sequence (Wu et al., 2022), as well as applying DP-SGD only to

certain selected tokens that are considered sensitive (Shi et al., 2022a).

The subcategory of fine-tuning language models with DP has seen even more research in the past few years (Mattern et al., 2022a; Shi et al., 2022b; Xu et al., 2021a; Basu et al., 2021; Petren Bach Hansen et al., 2022; Mireshghallah et al., 2021; Jana and Biemann, 2021; Vu et al., 2020; Qu et al., 2021; Bommasani et al., 2019; Li et al., 2022; Yu et al., 2022). The primary idea here is to take a pre-trained checkpoint of a model that was prepared using a public corpus of data (e.g. English Wikipedia and the BookCorpus (Zhu et al., 2015), as for the BERT model), and then fine-tune it on a sensitive dataset with guarantees of differential privacy. The fine-tuned DP model is then used for a given downstream task, such as text classification in the financial setting (Basu et al., 2021), sequence tagging (Jana and Biemann, 2021), synthetic text generation (Vu et al., 2020; Mattern et al., 2022a), and general NLP benchmarks such as datasets from GLUE (Wang et al., 2019) (Qu et al., 2021; Shi et al., 2022b; Yu et al., 2022).

The next large trend of differentially private NLP is that of **differentially private model representations**. In contrast to pre-training and fine-tuning language models with differential privacy, here the investigation is focused on the privatization of a particular latent representation of text, either at the document- or word-level (e.g. privatized word embeddings). These privatized representations can then be utilized by other models, such as privatized word embeddings used as input to a downstream model, or by further layers in the same model, due to the *closed under post-processing* property that DP provides, as described in Section 2.2.1.

One of the earlier studies to work in this setting is SynTF (Weggenmann and Kerschbaum, 2018), which is a system of creating synthetic term frequency (tf) (Luhn, 1957) and term frequency - inverse document frequency (tf-idf) (Sparck Jones, 1972) vectors by incorporating the exponential mechanism for an ε-DP guarantee, with the goal of preventing authorship attribution. Following this were a variety of different private text representation methods, including different types of private word embeddings (Xu et al., 2021b; Feyisetan and Kasiviswanathan, 2021; Xu et al., 2020b; Feyisetan et al., 2020, 2019; Carvalho et al., 2023; Plant et al., 2021; Xie and Hong, 2022; Carvalho et al., 2021), document-level representations (Meehan et al., 2022; Beigi et al., 2019; Lyu et al., 2020a) including privatized model encoders (e.g. Maheshwari et al. (2022)), as well as other types of representations, such as Latent Dirichlet Allocation (LDA) models (Blei et al., 2003) (Huang and Chen, 2021).

There is one important point that must be stressed for the word-level privatization techniques above. While the individual word embeddings are privatized with a certain DP guarantee, this does not translate to a full *document-level* privatization, in which an entire document is protected with that same DP guarantee. If we perturb every word of a given document with some noise addition mechanism to the original word vector, *the order of words is not taken into account*, meaning that syntactic information can theoretically still be revealed about the original text. This may be concerning for protection against attacks such as authorship attribution, in which the specific choice of word combinations may play a significant role in determining the author of a text. We discuss these points further in Section 5.2

of Chapter 5, as well as Section 6.2 of Chapter 6. See also Mattern et al. (2022a) for further discussion.

Another important aspect of some of the above studies (e.g. Feyisetan et al. (2019); Xu et al. (2021c)) is that the exact form of differential privacy in which they work is not the same as the classic form of DP outlined in Section 2.2.1. Specifically, Chatzikokolakis et al. (2013) proposed a generalization of DP for metric spaces, which is *Metric DP*, also known as $d_{\mathcal{X}}$-privacy. This is a form of DP that makes a modification to the guarantee outlined in Eqn. 2.1, multiplying the ε value by a *distance metric* $d(x, x')$, with d being the distnace function $d : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}^+$ and $w, w' \in \mathcal{X}$ being a pair of inputs from the discrete domain $\mathcal{X}$. This is illustrated in Def. 2.2.11.

Definition 2.2.11. *For $\varepsilon \geq 0$ and a distance metric $d : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}^+$, a mechanism $\mathcal{M}$ is εd-differentially private if, for all $S \subseteq Range(\mathcal{M})$, and for any $x, x' \in \mathcal{X}$, the following holds true:*

$$\Pr[\mathcal{M}(x) \in S] \leq e^{\varepsilon d(x,x')} \Pr[\mathcal{M}(x') \in S] \tag{2.29}$$

It is clear that, given various metrics d on $\mathcal{X}$, the strength of the privacy guarantee will vary. Note that we achieve the classic local DP setting if, for all $x \neq x'$, we set $d(x, x') = 1$. Importantly, the final reported ε value in a study utilizing metric DP is *not comparable* to the standard local DP setting. Since these distance metrics can vary from study to study, it is crucial to be clear on exactly how a given metric DP guarantee can convert to a local DP guarantee, i.e. what is the scaling factor that any ε in the local DP setting is being multiplied by in the metric DP setting. Since ε is defined as an exponent, given that $d(x, x') > 1$, the resulting local DP guarantee will be exponentially weaker, recalling Eqn. 2.3. In contrast, $d(x, x') < 1$ would provide a stronger guarantee than in the local DP setting.

The third research trend in differentially private NLP is that of **synthetic data generation with differential privacy**. This is a smaller but emerging subfield in which the primary goal is to output a privatized text representation in the form of discrete tokens, as opposed to latent representations. This is often performed in the setting of local differential privacy (e.g. Bo et al. (2021)), although in some cases also in the global DP setting (e.g. Mattern et al. (2022a)).

For instance, Bo et al. (2021) present a variant of the exponential mechanism, which they term the two-set exponential mechanism, using a seq2seq autoencoder. In their system, a bidirectional GRU (Cho et al., 2014a), acting as the encoder, is used to obtain a latent vector representation of an input text, with a generator GRU calculating logit weights for each token at a given time step. The two-set exponential mechanism is then used with the logits from this trained generator to privately sample individual tokens, in order to rewrite a given input sequence.

Another study, proposing the ADePT model (Krishna et al., 2021), also looked into rewriting an input text sequence using an RNN-based autoencoder. In contrast

to Bo et al. (2021), they utilized a Laplace and Gaussian mechanism over a latent representation of the full input document, decoding from this perturbed representation. We discuss this model in more detail in Chapters 5 and 6. Other work that has tackled synthetic data generation, either from the perspective of rewriting input texts, or for settings such as synthetic dataset creation, include (Yue et al., 2021; Wang et al., 2021; Xu et al., 2021c; Qu et al., 2021; Feyisetan et al., 2021; Bommasani et al., 2019).

As with the case of several of the above studies relating to differentially private text representations, here the same two points regarding word-level DP and metric DP are also valid. So far, the majority of the above studies (Yue et al., 2021; Bo et al., 2021; Wang et al., 2021; Xu et al., 2021c; Qu et al., 2021; Feyisetan et al., 2021) investigate private text generation at the granularity of words, as opposed to larger units, such as documents. Among these, metric DP has also been used (Yue et al., 2021; Xu et al., 2021a; Qu et al., 2021), in contrast to the classic DP framework, outlined in Section 2.2.1.

Apart from the final category of DP research in NLP, outlined below, there are a few additional studies that do not quite fit into the above three categories. These include combined methods of differential privacy and federated learning (McMahan et al., 2017), such as (Qi et al., 2020, 2021; Thakkar et al., 2021), as well as studies that use another framework as their primary subject of study, with differential privacy only as a comparison method or baseline (Lee et al., 2022; Mireshghallah et al., 2021).

The final research trend consists of **general discussion and analysis** studies. These are studies that analyze research directions for one of the other categories above, discuss general trends and potential shortcomings of existing methods, and suggest future directions to follow. One such study is Feyisetan et al. (2021), looking into challenges of DP text generation mechanisms and how to deal with improving the privacy/utility trade-off, reducing the high amount of noise that needs to be added to text representations in the DP setting. They propose some frameworks as possible solutions, such as deferring noise to a privacy amplification step. Similarly, Bommasani et al. (2019) examine strategies for synthetic text generation with DP.

Another set of studies is Habernal (2021) and Habernal (2022), investigating recent methods that have been proposed in the differentially private NLP literature (Krishna et al., 2021; Beigi et al., 2019), demonstrating some confusion of these methods with regards to their claimed DP guarantees. Habernal (2022) additionally proposes an algorithm to use as an empirical sanity check, in order to demonstrate whether a given DP mechanism actually violates its DP claims.

Overall, this is related to the wider problem that is present in some DP research investigations, in which crucial aspects of the differentially private mechanism are not made clear. This includes the exact epsilon values that are used, the unit of privatization (e.g. document-level DP, user-level DP, and so forth), as well as various details on the exact operations of the proposed DP mechanism. We discuss this in more detail in Chapter 5.

Finally, Mattern et al. (2022a) point out the flaws in DP mechanisms that operate at the word level, as discussed above, while Klymenko et al. (2022) discuss the current trends in differentially private NLP, possible vulnerabilities of existing methods, as well as important unsolved problems (e.g. explainability of DP methods, dealing with the unstructured nature of language) and future directions.

With regards to the current thesis, we provide contributions into several of the above categories. First, in Chapters 3 and 4, we work largely in the first line of research, investigating strategies for differentially private training of models used in NLP, including extensive study of some common language models in the DP-SGD setting (BERT (Devlin et al., 2019) and XtremeDistilTransformer (Mukherjee et al., 2021)). For Chapters 5 and 6, we move our focus to the third line of research, dealing with the problem of differentially private text generation. This is also related to the second line of research, in which we investigate effective internal textual representations of transformer models. Finally, throughout this thesis, we address questions related to the last line of research, including how to deal with strict noise requirements in the DP setting for textual data, as well as what exactly is the object of privatization for DP applications in the NLP domain.

2.3 Relevant NLP Concepts

Before moving on to tackling the main research questions of this thesis with respect to privacy and Natural Language Processing (NLP), we first present some essential concepts that are fundamental in the field of NLP. This includes the current standard procedures used in NLP, the primary tasks that are carried out as the goal of an NLP system, as well as common architectures that are used for this process.

2.3.1 Background

We describe below the standard structure of a contemporary NLP system, largely based off of methodologies common in modern-day machine learning. The overall idea is that we develop a model that produces a probability distribution, given some input data (e.g. raw text), in order to make predictions on the data, such as classifying it into a specific category (e.g. positive or negative sentiment of a text). In the current machine learning and NLP sphere, this model in practice is a deep neural network (DNN) architecture, discussed in Section 2.3.3. To prepare such a model, we first need to train it on the given task, such as text classification. This is typically performed by minimizing a *loss function*, given the model's predictions and some gold labels that represent the correct answer for a set of data points to the prediction problem. We therefore need a *training* dataset D_{train} for the model to learn from, containing these pairs of input data points and their corresponding labels, $(x_1, y_1), \ldots, (x_n, y_n)$, for a dataset of size n. Typically, $x_i \in \mathbb{R}^d$ is a feature vector of dimension d, such as a vectorized representation of a textual token or document, discussed in more detail in Section 2.3.3. To actually learn from this training set, we need a way to estimate how well the model carried out its prediction. Let us represent the model as a function $f_\theta(x)$ that takes in input data points x and is parameterized by a set of parameters θ. Using a loss function l, for a given prediction

and ground-truth label pair $(f_\theta(x_i), y_i)$, where $f_\theta(x_i)$ is the model's prediction on a data point x_i, we can calculate how well our model performs on the full training set as follows:

$$\mathcal{L}(\theta, D) = \frac{1}{n} \sum_{i=1}^{n} l(f_\theta(x_i), y_i) \tag{2.30}$$

To train our model, the goal is thus to minimize this loss, finding the optimal parameters $\hat{\theta}$ that result in the lowest $\mathcal{L}(\theta, D)$. By far the most common method to achieve this in machine learning is through the *gradient descent* algorithm (Cauchy et al., 1847). The gradient of the loss function $\nabla_\theta \mathcal{L}(\theta, D)$ is the vector of partial derivatives that represents how a change in each individual model parameter θ_j affects the loss function output. The general procedure of the algorithm is then as follows. At a given time step t, we pass our input data through the model, obtain predictions, and calculate the resulting loss. We then calculate the gradient of this loss and use it to update our parameters θ by taking a step in the opposite direction of it, controlling the step size with a learning rate $\gamma \in \mathbb{R}^+$. This is represented in Eqn. 2.31.

$$\begin{aligned} g_D &= \nabla_{\theta_t} \mathcal{L}(\theta_t, D) \\ \theta_{t+1} &= \theta_t - \gamma g_D \end{aligned} \tag{2.31}$$

We then iteratively repeat this procedure until we have reached a local minimum for $\mathcal{L}(\theta, D)$.

In practice, in models such as deep neural networks, at each time step we form a mini-batch of examples B taken from the training set, minimizing the loss using mini-batch stochastic gradient descent (SGD) (which can be traced back to Robbins and Monro (1951)), as in Eqn. 2.32.

$$g_B = \frac{1}{|B|} \sum_{i=1}^{|B|} \nabla_\theta \mathcal{L}(\theta, B) \tag{2.32}$$

The specific type of loss function used can vary depending on the problem, but a common choice in machine learning is the *cross-entropy loss*, as in Eqn. 2.33.

$$\begin{aligned} H(p, q) &= -\mathbb{E}_p[\log q] \\ &= -\sum_{x \in \mathcal{X}} p(x) \log(q(x)), \end{aligned} \tag{2.33}$$

for probability distributions p and q, with the same support $\mathcal{X}$. Relating to the notion of entropy from information theory, this loss function calculates the average number of nats (or bits, if working with base 2 logarithms) that are required to encode some data that is generated from a distribution p, when using a model q. In our case, $p(x)$ is the probability of the true class label that appears in the dataset, being 1 for the target label's class and 0 for all other classes. $q(x)$ then defines the probability that our model assigns to each label. The more similar the predicted probability distribution from model q to the true distribution p, the lower the cross

entropy loss. To obtain the cross entropy loss for the entire training set or given mini-batch, we typically calculate the average cross entropy over all data points in the set.

After having trained our model through the above optimization procedure, the result is a parameter vector $\hat{\theta}$ which allows the model to correctly perform on the given task, such as text classification. As discussed in Section 2.2.4, this parameter vector is obtained by learning information from the training set D_{train}, meaning that it can theoretically be exploited to reconstruct the original data points (Fredrikson et al., 2014, 2015; Hitaj et al., 2017; Shokri et al., 2017). In order to protect this information, we therefore need to privatize our model if it is to be released, using techniques presented above such as DP-SGD.

Finally, in order to properly evaluate performance, we use a separate *test* dataset D_{test} that contains another set of data points with their gold labels, not present in the training dataset. This is then used to evaluate how well our model has learned the prediction task, and whether it is able to generalize to examples unseen from the training set. For proper optimization of hyperparameters, we also require a *validation* dataset partition $D_{validation}$, separate from both the training and test sets.

2.3.2 Tasks

We briefly list some common NLP tasks that will be relevant for this thesis.

Text Classification

The task of *text classification* forms a major branch of NLP, which deals with the procedure of categorizing text into precise groups. The primary goal is to design a model that can learn to predict a label, given a sequence of textual tokens. There are a large number of specific tasks that fall under the blanket term of text classification. One of the most popular is *sentiment analysis*, in which the associated label with a given text represents the emotional tone of the author. This is commonly the polarity of the text, such as whether it is positive or negative. A very common dataset for this task is from Maas et al. (2011), with approximately 50,000 movie reviews from the Internet Movie Database (IMDb), each associated with a positive or negative label. Overall, the main idea of text classification is that we can take raw, unstructured text, and classify it into pre-determined categories, in order to be able to make a clearer sense of it.

Natural Language Inference (NLI)

This is another classification task, this time on pairs of text sequences. The first text sequence is referred to as a *premise*, while the second is a *hypothesis*. We proceed to classify the relationship between these two, which is generally divided into three classes. The first is *entailment*, in which it is possible to infer the hypothesis from the premise. A *contradiction* is when it is possible to infer the negation of the hypothesis from the premise. For any other case, we classify the relationship as

neutral. Common datasets for NLI include the Stanford Natural Language Inference (SNLI) Corpus (Bowman et al., 2015), as well as the Multi-Genre NLI (MultiNLI) Corpus (Williams et al., 2018). The former contains 570k sentence pairs, which are annotated with information about textual entailment, while the latter contains 433k such sentence pairs, but from ten different genres (e.g. Fiction, Letters, and so forth).

Named-Entity Recognition (NER)

In contrast to the above two tasks, NER is a *sequence labeling* task. This is also a type of classification task, but instead of classifying entire sequences, here *individual elements* of the sequence are classified. For NER, the goal of a model is to predict the named entity category of elements in a sequence. This includes tags such as *place*, *organization*, *time*, and *location*.

A common labeling scheme for NER datasets is the IOB tagging format (Ramshaw and Marcus, 1995), with a slight variation being the IOB2 format (Ratnaparkhi, 1998). In IOB2, a 'B-' prefix before the specific NER tag indicates that the tag is for a token at the beginning of a phrase. An 'I-' prefix represents the inside of a phrase, while an 'O' tag is for a token that is not part of any phrase. For example, the sentence "Fred went to San Marino" would be tagged as follows:

Fred	B-PER
went	O
to	O
San	B-LOC
Marino	I-LOC

Here, 'PER' refers to a *person* tag, while 'LOC' is a *location* tag.

The most common dataset in the NLP community for NER is the CoNLL-2003 shared task dataset (Tjong Kim Sang and De Meulder, 2003). It contains approximately 14,000 sentences in its training set, with around 300,000 tokens in total.

Part-of-Speech (POS) Tagging

As with NER, POS tagging is also a sequence labeling task. It is the process of assigning a grammatical tag to each token in a given sequence. In addition to common parts of speech such as *nouns*, *verbs*, and *adjectives*, many other categories are included, depending on the specific tagging system used. The two datasets investigated in Chapter 4, the Georgetown University Multilayer (GUM) Corpus (Zeldes, 2017) and the Universal Dependencies English Web Treebank (UD EWT) corpus (Silveira et al., 2014), utilize the Universal Dependencies set of POS tags (Petrov et al., 2012; Nivre et al., 2020). This system includes a total of 17 tags, such as 'DET' for a determiner, 'PRON' for a pronoun, 'ADP' for an adposition, and so forth.

Question Answering

For the task of question answering, the goal of the model is to provide an answer in the form of natural language, given a posed question. There are different variations for specifically carrying out this task. One of the most common is *extractive question answering*, in which a context is provided for the model. The model then needs to select an answer span from this context.

Another variation of this task is *generative question answering* (e.g. Yin et al. (2016); Lewis and Fan (2019)), in which the model directly generates the answer (see **Text Generation** below). For this latter type of QA, the generated answer can either be based on a provided context (open-book generative QA, e.g. Lewis et al. (2020b)), or without any context being explicitly provided (closed-book generative QA (e.g. Roberts et al. (2020)).

One of the most common datasets for this task is the Stanford Question Answering (SQuAD) dataset, which has seen multiple versions (SQuAD 1.1 (Rajpurkar et al., 2016) and SQuAD 2.0 (Rajpurkar et al., 2018)). The latter version of this dataset expands the original 100,000 answerable questions with an additional 50,000 unanswerable questions.

Text Generation

Another large branch of NLP is *text generation*, or *natural language generation* (NLG). In NLG, the main goal is for the NLP system to produce coherent and fluent natural language output, given some forms of textual input or meaningful representations (e.g. images, database of information, and so forth). There are many different tasks within NLG, including machine translation (MT) (e.g. Bahdanau et al. (2014); Vaswani et al. (2017)), image captioning (e.g. Xu et al. (2015)), as well as dialogue systems (Li et al., 2016; Zhang et al., 2020).

We tackle text generation in Chapters 5 and 6, specifically with the task of *text rewriting* using an autoencoder model, in which the goal of a model is to generate an output as close as possible to the input, reconstructing the original text sequence. By introducing a differentially private mechanism into such a model, we aim to rewrite an original text as closely as possible, but with provided privacy guarantees in the setting of local differential privacy.

Further Tasks

There are many tasks apart from the above that appear throughout the field and may be more problem-specific. This includes some of the tasks presented in Chapter 3 for graph datasets, such as predicting the category of a particular research paper, given a network of research citations.

NLP Tasks and the Importance of Privacy

It is important to note that, for all of the above tasks, there may be sensitive private information in the input texts. For example, users of online platforms providing

anonymous reviews for a product or service can be de-anonymized by means of various methods (e.g. Fabien et al. (2020)). This poses potential risks for the users such as retaliatory actions from the businesses that are being reviewed. In the case of question answering systems, a user may ask a sensitive question that reveals information about themselves, such as a medical condition or financial status. With more such sensitive information within a textual dataset, the need for obscuring this information with respect to each individual data provider becomes more pressing. It is therefore crucial to be able to design a privacy protection mechanism, either to be applied on the models that learn from such data, to restrict them from learning individual data points, or to be applied on the data itself, as in the case of privatized text rewriting.

2.3.3 Architectures

To conclude our discussion on the general non-private NLP setting, we present an outline of the common deep neural models that are utilized in NLP, that will be relevant for the rest of this thesis.

Artificial Neural Networks (ANNs)

In modern-day machine learning, by far the most common type of computational system is that of the *artificial neural network (ANN)*, which can be traced back to work by Rosenblatt (1958), inspired by even earlier work of McCulloch and Pitts (1943). This type of model is a collection of units, also known as artificial neurons, which process numerical input information by multiplying each dimension of the input signal by a learned weight parameter, taking the sum over these input-weight products, adding a learned bias parameter, and passing the result through a non-linear activation function, such as a sigmoid function, $\sigma(x) = \frac{1}{1+e^{-x}}$, or rectified linear unit (ReLU) function, $\text{ReLU}(x) = \max(0, x)$. These weights can be interpreted as a representation of the importance of information from a given input signal's dimension. This procedure produces an output neuron, and repeating this process over the input data multiple times creates a full layer of such neurons. In order to create more abstract representations of the original input data, multiple such neuron layers can be stacked in order to create a *feedforward neural network*, with a final output layer that is used for the model's prediction. All layers apart from the output and input are referred to as *hidden layers*. When a network has many such hidden layers, it is referred to as a *deep neural network (DNN)*. Formally, a simple one-layer feedforward neural network can be represented as in Eqn 2.34, with the first layer using a ReLU activation function, while the second using the sigmoid function.

$$f(\mathbf{x}; \mathbf{W_1}, \mathbf{W_2}, \mathbf{b_1}, \mathbf{b_2}) = \sigma(\mathbf{W_2}^\intercal(\text{ReLU}(\mathbf{W_1}^\intercal \mathbf{x} + \mathbf{b_1})) + \mathbf{b_2}), \tag{2.34}$$

where $\mathbf{x} \in \mathbb{R}^d$, $\mathbf{W_1} \in \mathbb{R}^{d \times h}$, $\mathbf{W_2} \in \mathbb{R}^{h \times k}$, $\mathbf{b_1} \in \mathbb{R}^h$, and $\mathbf{b_2} \in \mathbb{R}^k$, with d, h, and k being the input, hidden, and output dimensions, respectively.

As with all other models described below, the general training procedure is the same as outlined in Section 2.3.1. In the case of ANNs, in order to actually calculate

the gradient of the loss function with respect to the model's parameters, we use the *backpropagation* algorithm (Rumelhart et al., 1986), which can be seen as a way to efficiently apply the chain rule for each layer of the network, starting from the last layer.

The incorporation of ANNs into the field of NLP has led to rapid improvements in performance compared to previous statistical NLP systems, with a widespread adoption of such models throughout the different branches of the field (e.g. various text classification tasks, natural language generation, and so forth). We are thus currently in an era of *neural NLP*, with the majority of new methodologies proposed deeply rooted in ANNs.

Word Embeddings

The field of *distributional semantics* interprets language meaning based on the *distributional hypothesis*, popularized by Firth (1957) with the idea that the meaning of a word is represented by "the company it keeps", i.e. its distributional properties. This has become the most popular way to model word meaning within NLP, using **word embeddings**. These are representations of individual words as real-valued vectors, which encode their distributional information such that words with similar meanings, i.e. words that appear in similar contexts, will have corresponding vectors that are near one another in the vector space. Some very popular methodologies for preparing such vectors are the skip-gram and continuous bag-of-words (CBOW) models (Mikolov et al., 2013), collectively known as *word2vec*, the related fastText model (Joulin et al., 2017), as well as the GloVe ("Global Vectors") model (Pennington et al., 2014). In the case of *word2vec*, a two-layer feedforward neural network is used to either predict context words, given a target word as input (skip-gram), or predict a target word, given context words as input (CBOW), over a sliding window that is iterated over a textual corpus. Since the input and output words of this network are represented as one-hot vectors, the resulting dimension of the first weight matrix associated with each word becomes that word's embedding. The fastText model is an extension to *word2vec*, which operates at the character level, being able to construct vector representations for out-of-vocabulary (OOV) words. In the case of GloVe, a co-occurrence matrix $X \in \mathbb{R}^{V \times V}$ is created from a textual corpus of vocabulary V, with each X_{ij} representing how often word i co-occurred with word j. A log-bilinear model is then trained with a weighted least-squares objective, predicting the log ratio of co-occurrence probabilities from X, with the intuition that this ratio can represent word meaning in some form. All three types of word embeddings are examples of *dense* vector representations, which are relatively information-rich and mostly contain non-zero values, in contrast to sparse vectors, which tend to have far more dimensions, many of which are zeroes.

An important note on the above models is that they create *static* word embeddings, in which a single embedding is used to represent a word, regardless of polysemy or homonymy. To overcome this limitation, *contextual* word embeddings have been developed based on the notion of pre-trained language models (discussed under **Pre-trained Language Models (PLMs)** below), such as the ELMo ("Embeddings from Language Model") and BERT ("Bidirectional Encoder Representations

from Transformers") models. From these models, the resulting word embeddings are different for every occurrence of a word in a given sequence.

Recurrent Neural Networks (RNNs)

This is a family of neural networks that are designed to process sequential data, such as sequences of tokens in a textual dataset. The primary idea of RNNs is that the input is a sequence $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(T)}$, composed of T time steps. For each time step, a value in the sequence is input to the model, which then produces an output at that time step. Importantly, an additional weight matrix is introduced into the model that produces *recurrent connections* between hidden units, i.e. loops in the network which allow the output of units to affect the input to those same units. More formally, a given time step of an RNN can be represented as in Eqn. 2.35, for a simple one-layer RNN (Rumelhart et al., 1985).

$$f(\mathbf{x}^{(t)}; \mathbf{W_1}, \mathbf{W_2}, \mathbf{U_R}, \mathbf{b_1}, \mathbf{b_2}) = \sigma(\mathbf{W_2}^\intercal(\tanh(\mathbf{W_1}^\intercal \mathbf{x}^{(t)} + \mathbf{U_R}^\intercal \mathbf{z}^{(t-1)} + \mathbf{b_1})) + \mathbf{b_2}), \tag{2.35}$$

where $\mathbf{x} \in \mathbb{R}^d$, $\mathbf{U} \in \mathbb{R}^{h \times h}$ is the weight providing the recurrent connection, $\mathbf{W_1} \in \mathbb{R}^{d \times h}$, $\mathbf{W_2} \in \mathbb{R}^{h \times k}$, $\mathbf{b_1} \in \mathbb{R}^h$ and $\mathbf{b_2} \in \mathbb{R}^k$ are weight matrices and bias vectors, respectively, and $\mathbf{z}^{(t-1)} \in (-1, 1)^h$ is the hidden representation for the previous time step, i.e. $\mathbf{z}^{(t-1)} = \tanh(\mathbf{W_1}^\intercal \mathbf{x}^{(t-1)} + \mathbf{U_R}^\intercal \mathbf{z}^{(t-2)} + \mathbf{b_1})$. Note that the *tanh* activation function is often used for RNNs, where $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. In contrast to feedforward neural networks, a modified version of the backpropagation algorithm, namely *Backpropagation Through Time (BPTT)* (Werbos, 1988; Mozer, 1995) is used to obtain the gradient of an RNN model. Essentially, the RNN is treated as an unrolled, multi-layer DNN, to which backpropagation is applied.

Two notable issues with training simpler RNN architectures (e.g. Elman (1990)) are the *vanishing* and *exploding* gradient problems (Bengio et al., 1994; Pascanu et al., 2013). In the former case, the norm of the gradient rapidly decreases to 0 during the training process, resulting in difficulties for the model to learn connections between distant time steps. Similarly, in the latter case, the norm of the gradient rapidly increases exponentially. Repeated applications of the chain rule during backpropagation can result in very small values being multiplied iteratively to cause the vanishing gradients problem, effectively blocking weights from being changed in value. Alternatively, large values multiplied iteratively cause the exploding gradients problem. These two problems in fact also affect feedforward neural networks with many layers. Various techniques have been suggested to mitigate these issues, such as gradient clipping (Pascanu et al., 2013) and batch normalization (Ioffe and Szegedy, 2015). Interestingly, the gradient clipping strategy is also one of the core components of the DP-SGD algorithm, restricting the range of values that the gradient can take on, in order to bound the amount of required added noise to the gradient in the differentially private setting.

Special RNN architectures have also been designed to be less susceptible to the above two problems, and allow for modeling dependencies across more distant time steps. One such common architecture is the long short-term memory (LSTM) network (Hochreiter and Schmidhuber, 1997). The primary idea is to expand on

the structure of the simpler RNN models to include an LSTM unit, which contains a series of gates that control information flow. These are the *input gate*, regulating information flow into the LSTM unit at each time step, the *output gate*, regulating the information that is output from the unit, the *forget gate*, determining which information to not pass on to the next time step, as well as the *cell*, which acts as a 'conveyor belt' for information to flow across time steps. More formally, an LSTM unit can be defined as the following set of equations:

$$\begin{aligned}
\mathbf{i}^{(t)} &= \sigma(\mathbf{W_i}^\intercal \mathbf{x}^{(t)} + \mathbf{U_i}^\intercal \mathbf{z}^{(t-1)} + \mathbf{b_i}) \\
\mathbf{o}^{(t)} &= \sigma(\mathbf{W_o}^\intercal \mathbf{x}^{(t)} + \mathbf{U_o}^\intercal \mathbf{z}^{(t-1)} + \mathbf{b_o}) \\
\mathbf{f}^{(t)} &= \sigma(\mathbf{W_f}^\intercal \mathbf{x}^{(t)} + \mathbf{U_f}^\intercal \mathbf{z}^{(t-1)} + \mathbf{b_f}) \\
\tilde{\mathbf{c}}^{(t)} &= \tanh(\mathbf{W_g}^\intercal \mathbf{x}^{(t)} + \mathbf{U_g}^\intercal \mathbf{z}^{(t-1)} + \mathbf{b_c}) \\
\mathbf{c}^{(t)} &= \mathbf{f}^{(t)} \odot \mathbf{c}_{t-1} + \mathbf{i}^{(t)} \odot \tilde{\mathbf{c}}^{(t)} \\
\mathbf{z}^{(t)} &= \tanh(\mathbf{c}^{(t)}) \odot \mathbf{o}^{(t)},
\end{aligned} \tag{2.36}$$

where $\mathbf{x} \in \mathbb{R}^d$, $\mathbf{W} \in \mathbb{R}^{d \times h}$ and $\mathbf{b} \in \mathbb{R}^h$ are the input weight matrices and bias connections for the different gates, respectively, $\mathbf{U} \in \mathbb{R}^{h \times h}$ are the recurrent weight matrices, $\mathbf{i^{(t)}}, \mathbf{o^{(t)}}, \mathbf{f^{(t)}} \in (0, 1)^h$ are the input, output and forget gates at time step t, respectively, $\tilde{\mathbf{c}}^{(t)} \in (-1, 1)^h$ is the new cell input at time step t, $\mathbf{c}^{(t)} \in \mathbb{R}^h$ is the cell state, and $\mathbf{z}^{(t)} \in (-1, 1)^h$ is the hidden state and output of the LSTM unit.

By regulating the flow of information in and out of the LSTM unit using these gates, it is possible to preserve useful information across further time steps and thereby achieve superior performance on tasks associated with sequential data, in comparison to the simpler RNN architectures. One further common extension to recurrent networks is the *bidirectional RNN* (Schuster and Paliwal, 1997), in which two recurrent hidden layers are combined, with sequential information being input to each layer in opposite directions. This type of RNN thus learns both past and future information from the sequence simultaneously, which may help when the model benefits from more contextual information from the full sequence.

With regards to training RNN-type models, an additional methodology may be employed known as **teacher forcing**. During the training process, at each time step t, the model receives the true label $y^{(t-1)}$, as opposed to the model's own hidden state output $\mathbf{z}^{(t-1)}$. This may be useful in training the model by forcing it to more closely replicate the input sequence. However, this could also cause problems for the model at test time, since such ground-truth labels would not be available. Therefore, it is possible to find a balance between the two training regimes, only using teacher forcing with a certain probability p at each iteration.

Within the field of NLP, RNN-based models have been very popular (e.g. Sutskever et al. (2014); Cho et al. (2014b); Bahdanau et al. (2014); Peters et al. (2018)), in part due to their sequential nature being an inherent good fit for textual data. At each time step, an RNN model outputs a hidden vector $\mathbf{z}^{(t)}$, with a final hidden vector output $\mathbf{z}^{(T)}$, at the end of a sequence of T time steps. Since this final vector contains a summary of the full input sequence, it can naively be used for a sequence classification task, for example adding a final feedforward layer that transforms this

hidden vector into a probability distribution over k possible classes. Similarly, it can also be used for a sequence tagging task (e.g. named-entity recognition) by utilizing the hidden vector $\mathbf{z}^{(t)}$ at every time step, input to a feedforward layer for classification of individual tokens in the sequence.

Despite the successes of RNNs, long-term dependencies still remain a problem for this family of models. While enhancements such as LSTM units are able to mitigate the issue to some extent, the recurrent nature of the model still leads to a limitation on the long-range information that can be captured. Additionally, there is also the issue of parallelization, where RNN-based models require information to be processed token-by-token, which means that they cannot be trained in parallel. Among the various solutions to these issues (e.g. Bai et al. (2018)), by far the most significant has been the proposed Transformer model, that has become overwhelmingly dominant in the NLP community, discussed in more detail below under **Transformers**.

Sequence to Sequence Models

In the case of text generation tasks, the size of the output sequence may be different from that of the input sequence. For example, if we wish to translate from a source language to a target language (e.g. English to French), the two sequences would not be expected to be of the same size. The family of models known as **sequence to sequence**, or **seq2seq** models deals exactly with such tasks. The most common type of architecture for these models is the **encoder-decoder** architecture, which uses two separate models, one for *encoding* the source sequence, the other for *decoding* from the outputs of the encoder into the target sequence.

Given a sequence of inputs $(x_1, \ldots, x_T)$, we wish to compute a sequence of outputs $(y_1, \ldots, y_{T'})$, where T and T' are not necessarily the same. A simple way to achieve this is to utilize one LSTM model as an *encoder*, operating on the input sequence, while another LSTM is used as a *decoder*, taking the final hidden state of the first LSTM, $\mathbf{z}^{(T)}$, and setting it as its initial hidden state, as in Sutskever et al. (2014) and Cho et al. (2014b).

One significant issue in this setup, is that the final hidden state of the first LSTM, $\mathbf{z}^{(T)}$, has the burden of capturing information from the entire input sequence $(x_1, \ldots, x_T)$. This is especially prominent with longer input sequences, where performance would be expected to significantly deteriorate. In order to overcome this issue, Bahdanau et al. (2014) proposed an enhancement to the encoder-decoder architecture, introducing an **attention** mechanism. In essence, this allows the decoder to selectively look through the encoder outputs for each token of the input sequence, as opposed to relying on only one fixed vector. We can term this context vector used by the decoder as $\mathbf{c}^{(t')}$, for time step t' of the output sequence, replacing the previous $\mathbf{z}^{(T)}$. As the decoding process continues at each time step of the output sequence, this vector is updated, based on the encoder hidden states $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(T)}$ and the decoded sequence up to time step $t' - 1$ $(\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(t'-1)})$. More specifically, an *alignment model* is prepared, as in Eqn. 2.37, which predicts a score between a

hidden vector of the input sequence and the previous time step of the decoder.

$$e_{t't} = a(\mathbf{s}^{(t'-1)}, \mathbf{z}^{(t)}), \tag{2.37}$$

representing the score between time step $t'-1$ of the decoder and time step t of the encoder. This function a is simply another feedforward neural network, trained as part of the rest of the full encoder-decoder model. Obtaining each $e_{t't}$, we can then compute a weight for this alignment using the softmax function, as in Eqn. 2.38.

$$\alpha_{t't} = \frac{\exp(e_{t't})}{\sum_{k=1}^{T} \exp(e_{t'k})} \tag{2.38}$$

The final context vector $\mathbf{c}^{(t')}$ at time step t' of the decoder is then computed as a weighted sum of the encoder output hidden states, as in Eqn. 2.39.

$$\mathbf{c}^{(t')} = \sum_{t=1}^{T} \alpha_{t't} \mathbf{z}^{(t)} \tag{2.39}$$

With the performance improvements of the above attention mechanism (termed *Additive Attention*), other forms of attention were explored in subsequent work. Notably, Luong et al. (2015) proposed alternatives to calculating the alignment scores, as well as distinguishing between *global* and *local* attention, which operate on the full source sequence and a smaller subset of it, respectively. Importantly, the Transformer model, itself a type of encoder-decoder network, was developed based on the related notion of *self-attention*, outlined below.

Transformers

The Transformer model (Vaswani et al., 2017) is a sequence to sequence encoder-decoder model that was originally proposed for the task of machine translation. In contrast to the RNN-based models above, the Transformer does not use recurrent connections, instead primarily relying on attention. By avoiding recurrence, it is able to achieve far more parallelization. Since its appearance, it has become arguably the most prominent architecture in the NLP community, with many prevalent models that subsequently appeared being derived from it (e.g. see **Pre-trained Language Models (PLMs)**).

The Transformer consists of multiple stacked encoder and decoder 'blocks' within its encoding and decoding components, respectively, with six each in the original version of the model. It takes a sequence of inputs $\mathbf{x} = (x_1, \ldots, x_T)$ and outputs a sequence $\mathbf{y} = (y_1, \ldots, y_{T'})$. The encoder blocks contain two subcomponents, *self-attention* and a feedforward neural network. The primary goal of this self-attention layer is to allow the model to look at representations of all tokens in the input sequence, as it is processing every individual token. It is therefore one of the mechanisms that allows to bypass the necessity of the recurrent connections in RNNs, which provide the network with information on previous tokens that the model has seen, apart from the target token. Interestingly, the majority of the Transformer's weights are present in the subsequent feedforward neural network layer.

Hence, although there is less discussion about it in the literature, it contains a lot of the Transformer's representational power. The decoder blocks then contain three subcomponents, being self-attention, the feedforward neural network, as well as a *cross-attention* layer in-between them, which allows the decoder block to attend over parts of the outputs of the encoder.

The self-attention mechanism itself works as follows. The overall process is similar to other types of attention, outlined in Section 2.3.3, with (1) an alignment model, (2) softmax step, as well as (3) a weighted summation step. In encoder-decoder attention (e.g. Bahdanau et al. (2014)), each decoder element is put through the alignment model to obtain an alignment score with all encoder elements. In contrast, in encoder self-attention, each element of the encoder is scored with each other element of the encoder. Analogously, in decoder self-attention, each element of the decoder is scored with each other element of the decoder.

The alignment model is labeled *scaled dot-product attention* by Vaswani et al. (2017) and is computed as in Eqn. 2.40.

$$e_{ij} = \frac{(\mathbf{W}_q^\intercal \mathbf{z}_i)^\intercal (\mathbf{W}_k^\intercal \mathbf{z}_j)}{\sqrt{d_k}}, \tag{2.40}$$

where $\mathbf{W}_q \in \mathbb{R}^{d_z \times d_q}$ and $\mathbf{W}_k \in \mathbb{R}^{d_z \times d_k}$ are weight matrices used to transform the output of the previous encoder or decoder block, $\mathbf{z}_i \in \mathbb{R}^{d_z}$, for a given token position i, where d_z is the dimension of the previous layer output. Each input token $\mathbf{z}_i$ to this alignment model will have a resulting *query* representation of dimension d_q and an output *key* representation of dimension d_k, after multiplication by the $\mathbf{W}_q$ and $\mathbf{W}_k$ weight matrices, respectively, where $d_q = d_k = d_z$. The scaling factor of $\frac{1}{\sqrt{d_k}}$ is introduced to deal with gradient stability when using larger values of d_k.

The softmax step is performed similar to Eqn. 2.38, shown in Eqn. 2.41.

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T} \exp(e_{ik})}, \tag{2.41}$$

where T is the length of the input sequence to the self-attention layer.

Finally, the weighted sum step is performed similar to Eqn. 2.39. We take the same input vector to the self-attention mechanism for a given token position i, $\mathbf{z}_i$, this time multiplying it with a weight matrix $\mathbf{W}_v \in \mathbb{R}^{d_z \times d_v}$ to create a *value* representation of dimension $d_v = d_z$. A weighted sum of these value representations are then computed by multiplying them with the resulting self-attention scores from Eqn. 2.41, as in Eqn. 2.42.

$$\mathbf{z}_i^{'} = \sum_{j=1}^{T} \alpha_{ij} \mathbf{W}_v \mathbf{z}_j \tag{2.42}$$

We subsequently have the output of the self-attention mechanism $\mathbf{z}_i^{'}$, for each token position i. We define the above three steps of self-attention in a more compact manner in Eqn. 2.43.

$$\text{Attention}(\mathbf{Z}; \mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v) = \text{softmax}\left(\frac{(\mathbf{Z}\mathbf{W}_q)(\mathbf{Z}\mathbf{W}_k)^\intercal}{\sqrt{d_k}}\right)(\mathbf{Z}\mathbf{W}_v), \tag{2.43}$$

where $\mathbf{Z} \in \mathbb{R}^{T \times d_z}$ is the output of the previous encoder or decoder block.

An additional component added to this self-attention mechanism in the Transformer is *multi-headed* attention. In this case, the above attention computations are performed multiple times, with h different matrices for the query, key, and value weight matrices. Each of these computations is termed a *head*, with each head $\mathbf{Z}_i^{'}$ computed as in Eqn. 2.44.

$$\mathbf{Z}_i^{'} = \text{Attention}(\mathbf{Z}; \mathbf{W}_{qi}, \mathbf{W}_{ki}, \mathbf{W}_{vi}), \tag{2.44}$$

where $\mathbf{Z}_i^{'} \in \mathbb{R}^{T \times d_v}$. In order to maintain dimensionality of representations $\mathbf{Z}$ into, and out of, this multi-head self-attention module, the final outputs of all attention heads are concatenated and multiplied by another weight matrix, $\mathbf{W}_o \in \mathbb{R}^{hd_v \times d_z}$, which projects them back into $\mathbb{R}^{T \times d_z}$, as in Eqn. 2.45.

$$\text{MultiHead}(\mathbf{Z}_1^{'}, \ldots, \mathbf{Z}_h^{'}; \mathbf{W}_o) = \hat{\mathbf{Z}} = [\mathbf{Z}_1^{'}, \ldots, \mathbf{Z}_h^{'}]\mathbf{W}_o, \tag{2.45}$$

where $\hat{\mathbf{Z}} \in \mathbb{R}^{T \times d_z}$ is the final output of the multi-head attention module. The intuition behind using multi-head attention is that each individual attention head is able to attend to different patterns of information from the input sequence to the layer, overall reaching a richer interpretation of the sequence for the model.

Before passing on this $\hat{\mathbf{Z}}$ to the subsequent feedforward layer, a *residual connection* is introduced (He et al., 2016), followed by a *layer normalization* step (Ba et al., 2016). For the residual connection, the input to the layer is directly added to the output, i.e. $\hat{\mathbf{Z}} + \mathbf{Z}$. An intuition behind this, is that the residual connection allows gradients to flow through the network directly, avoiding issues such as gradient vanishing, leading to more robust training of the network.

In the case of layer normalization, let $\hat{\mathbf{z}}_i$ represent token position i of $\hat{\mathbf{Z}}$. Each $\hat{\mathbf{z}}_i$ is rescaled and recentered across dimension d_z, as in Eqn. 2.46.

$$\begin{aligned}
\boldsymbol{\mu}_i &= \frac{1}{d_z}\sum_{j=1}^{d_z} \hat{\mathbf{z}}_{ij} \\
\boldsymbol{\sigma}_i^2 &= \frac{1}{d_z}\sum_{j=1}^{d_z} (\hat{\mathbf{z}}_{ij} - \boldsymbol{\mu}_{ij})^2 \\
\hat{\mathbf{z}}_{i_{\text{norm}}} &= \frac{\hat{\mathbf{z}}_i - \boldsymbol{\mu}_i}{\sqrt{\boldsymbol{\sigma}_i^2 + \epsilon}} \\
\text{LayerNorm}(\hat{\mathbf{z}}_i; \boldsymbol{\gamma}, \boldsymbol{\beta}) &= \tilde{\mathbf{z}}_i = \boldsymbol{\gamma} \cdot \hat{\mathbf{z}}_{i_{\text{norm}}} + \boldsymbol{\beta},
\end{aligned} \tag{2.46}$$

where ϵ is a small constant introduced for numerical stability, while $\boldsymbol{\gamma} \in \mathbb{R}^{d_z}$ and $\boldsymbol{\beta} \in \mathbb{R}^{d_z}$ are learnable parameters during training, allowing the model to determine the mean and variance of the output distribution of values of the layer normalization step.

Following the self-attention layer, the feedforward neural network layer, here termed FFNN, is similar to the basic ANN described in Eqn. 2.34, with two linear

transformations and a ReLU non-linear activation function in-between. The overall procedure for each output token of the multi-headed self-attention mechanism in a transformer encoder block is thus as in Eqn. 2.47.

$$\begin{aligned} \tilde{\mathbf{z}}_i &= \text{LayerNorm}(\hat{\mathbf{z}}_i + \mathbf{z}_i) \\ \mathbf{y}_i &= \text{LayerNorm}(\text{FFNN}(\tilde{\mathbf{z}}_i) + \tilde{\mathbf{z}}_i), \end{aligned} \tag{2.47}$$

with another residual connection and layer normalization step following the feedforward neural network.

In a decoder block of the Transformer, an additional component is introduced between the multi-headed self-attention and feedforward neural networks, being *cross-attention*. The procedure is exactly the same as in self-attention, only here instead of operating on a single sequence, there is a comparison of the encoder outputs and the decoder's input sequence. This means that we can modify Eqn. 2.43, where the $\mathbf{Z}$ matrices associated with $\mathbf{W}_k$ and $\mathbf{W}_v$ are obtained from the output of the Transformer's final encoder block, while the $\mathbf{Z}$ matrix associated with $\mathbf{W}_q$ is from the decoder's own input sequence. In addition, the self-attention mechanism in a decoder block ensures that, during the decoding process at token position i, only positions less than i can be attended to by masking subsequent values. As with the self-attention and feedforward neural network sub-layers, the cross-attention sub-layer is also followed by a residual connection and a layer normalization step.

An additional point worth mentioning is the input to the very first encoder or decoder block. Each token is first put through a learned input embedding layer, which is basically just a feedforward linear layer that maps input tokens to a trained representation of hidden dimension d_z. Since the Transformer model described so far, unlike RNN-based models, does not have an inherent way to represent order within a sequence of tokens, an additional vector is added to each token's input embedding, described as the *positional encoding*. This provides the model with information on the relative or absolute position of each token in the input sequence, described in more detail in Vaswani et al. (2017).

Finally, the output of the final decoder block of the Transformer is followed by another feedforward linear layer, which projects the decoder block's final representation into a logits vector. This is then followed by a softmax layer to obtain the model's final probabilities over the vocabulary of the target sequence. During training, the Transformer is not autoregressive, predicting all outputs in parallel by using the true gold tokens of the output sequence, hence using teacher forcing. This parallelization is possible since all time steps are computed simultaneously, as opposed to the sequential computations of RNN-based models. At test time, we use autoregressive decoding, with the last predicted token of the decoder appended to the decoder inputs at each next decoding step, if using the greedy search algorithm. The overall architecture of the Transformer can be seen in Figure 2.8.

Pre-trained Language Models (PLMs)

A very prominent type of model within modern-day NLP is the **pre-trained language model (PLM)**. The vast majority of today's PLMs are based on the above

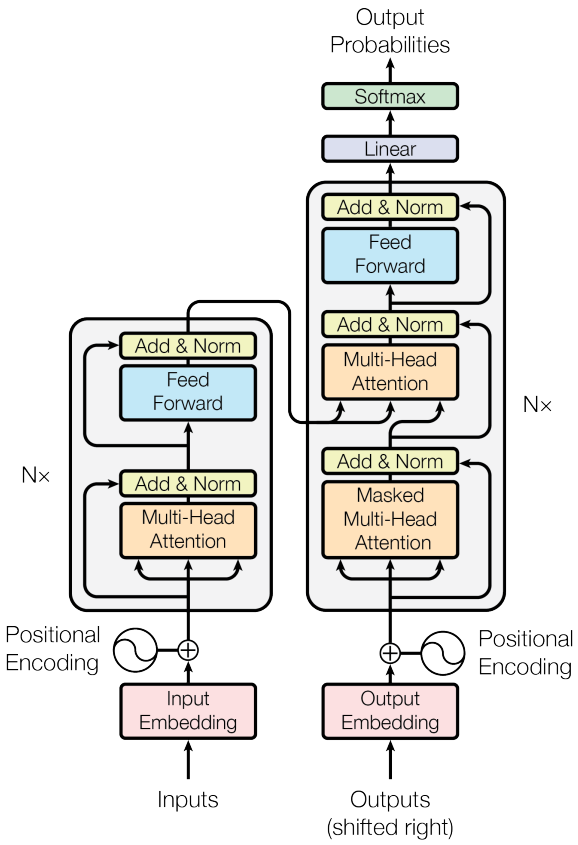


Figure 2.8: Transformer model architecture, diagram taken from Vaswani et al. (2017). Nx represents N such encoder-decoder blocks.

Transformer architecture. These models can either take only the encoder portion of the original Transformer model (Devlin et al., 2019; Liu et al., 2019; Sanh et al., 2019), only the decoder portion (Radford et al., 2019; Yang et al., 2019), or the full encoder-decoder structure (Lewis et al., 2020a; Raffel et al., 2020). There are a few PLMs also based on other neural architectures, such as LSTMs (e.g. ELMo of Peters et al. (2018)).

The primary goal is to learn useful language representations from a large unlabeled corpus of text, generally in a self-supervised manner. The typical tasks for pre-training a language model include next token prediction in a text sequence (e.g. Radford et al. (2019)), masked language modeling (e.g. Devlin et al. (2019); Liu et al. (2019)), as well as next sentence prediction (e.g. Devlin et al. (2019)).

The overall idea of PLMs is very closely related to that of *transfer learning*. After pre-training the language model on the self-supervised task, the learned language representations can then be transferred to a variety of downstream tasks, which would typically have a smaller amount of available data in the supervised setting. The pre-trained model representations can be fully *frozen*, i.e. with model parameters not updated for the downstream task, apart from an extra task-specific final classification layer added to the end of the model, replacing the final layer used for the self-supervised task during pre-training. Alternatively, the pre-trained model

representations may be *fully fine-tuned*, with all parameters being updated to learn the downstream task, as well as *partially fine-tuned*, in which only a specific subset of the model parameters are further optimized, such as the last few layers.

Overall, these models typically contain a large number of parameters, with the popular BERT model (Devlin et al., 2019) containing 110 million parameters in its `bert-base-cased` version,[3] which we use for our investigations in Chapter 4. More recently, large language models (LLMs) with parameters in the billions and higher (e.g. Brown et al. (2020)) have received a large amount of attention, both within the NLP community, as well as the general public, due to their considerable performance and capabilities, with continuous improvements in a variety of tasks, as the scale of the model is further increased (Wei et al., 2022).

Apart from the above set of architectures, there are many other types of neural networks that are used throughout machine learning and natural language processing. One such important network is the *Graph Neural Network (GNN)* and its most common variant, the *Graph Convolutional Network (GCN)*. These networks allow to extend the notion of ANNs to graph-based datasets and are presented in further detail in Chapter 3.

2.4 Chapter Summary

In this chapter, we have discussed the relevant background for the various research directions that are explored throughout this thesis. We started with the concept of *privacy from a philosophical perspective*, noting various theories that have been proposed over the last century and a half about how privacy can be defined. These include the control and limitation theories of privacy, as well as other types of distinctions on its categorization, such as normative vs. non-normative, and reductionist vs. non-reductionist views.

Next, we discussed *differential privacy (DP)*, the main framework that we are working in for the privatization of NLP systems. We motivated DP both from an informal, intuitive perspective, as well as the more rigorous mathematical definition, discussing the formal guarantees that it provides, as well as the various flavors of DP that exist. This includes pure $(\varepsilon, 0)$-DP vs. approximate (ε, δ)-DP and how to achieve them through the Laplace and Gaussian mechanisms, respectively, as well as global DP vs. local DP. We demonstrated the global DP setting by means of an example application of different DP mechanisms on a dataset of individuals with various attributes. We then discussed the randomized response technique, being the oldest DP algorithm and one of the most well-known examples of the local DP setting.

To proceed with our discussion of DP, we outlined how to apply the framework to machine learning, including a description of the DP-SGD algorithm and the Moments Accountant. We then concluded this section by presenting four primary research directions of applying DP to the NLP setting, including (1) pre-training

[3] Available at `https://huggingface.co/bert-base-cased`.

private language models, (2) privatizing internal model representations, (3) synthetic data generation with differential privacy, as well as (4) general discussion and analysis of the application of DP to the NLP domain.

Finally, we finish this chapter by presenting essential concepts from the field of NLP that will be relevant for the rest of this thesis. We discuss the standard structure of a modern NLP system and various common NLP tasks that are prominent throughout the NLP literature. We then conclude by describing in detail the primary deep neural models used in the field, starting from artificial neural networks, and finishing with transformers and pre-trained language models. We now move into the primary part of this thesis, starting with investigations into the privatization of text classification models with graph datasets in Chapter 3.

Chapter 3

Privatization of Graph Convolutional Networks for Text Classification

We now turn to tackling the first major set of questions in this thesis, namely the privatization of NLP models. We begin in this chapter with RQ1, originally defined in Chapter 1:

RQ1 How can we privatize text classification models that operate on graph datasets?

As we will see, this is far from a straightforward process, with difficulties of operating on graph datasets present even in the non-private scenario. The core of the problem lies in the fact that large graph datasets do not have a straightforward method to be divided into smaller subgraphs. This leads to significant memory overhead when training a model on a graph dataset without privacy, as well as a large amount of required added noise in the private setting, when using an algorithm such as DP-SGD. While DP-SGD can work mostly out of the box for more standard neural network models such as feedforward neural networks, and even Transformer models, applying it to the case of graph neural networks (GNNs) poses significant challenges. We show how to solve this problem using a graph splitting technique, attaining a good privacy/utility trade-off.

3.1 Graphs in Discrete Mathematics

A **graph** $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a fundamental structure within the field of graph theory in mathematics. It consists of two primary sets of objects, known as **vertices** or **nodes**, $\mathcal{V}$, and **edges**, $\mathcal{E}$. A vertex is the basic unit of a graph, with two vertices that are related in some way sharing an edge.

There are different types of graphs. One is the *directed* graph, in which there are orientations associated with the edges of the graph. This means that, for a given pair of nodes x and y, there may be an edge connecting them, such that the *head* of the edge is only on x, while the *tail* is only on y. We can thus represent this

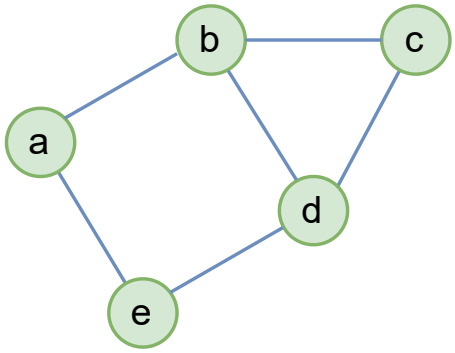


Figure 3.1: Diagram of a simple undirected graph with 5 vertices and 6 edges.

connection as an ordered pair, (x, y). Similarly, a graph may be *undirected*, lacking any orientations on its edges. In some cases, a graph can contain a *loop*, in which a given vertex has an edge that connects back to itself. The *degree* of a vertex is defined as the number of edges that are connected to it, with the *degree matrix* $\mathbf{D} \in \mathbb{N}^{n \times n}$ being a diagonal matrix that indicates the degree of each node, where $n = |\mathcal{V}|$ is the number of nodes in the graph. The *adjacency matrix* $\mathbf{A} \in \{0, 1\}^{n \times n}$ of a graph represents all edge connections between nodes, with $\mathbf{A}_{ij}$ representing the presence or absence of a connection from node i to node j. In *multigraphs*, two nodes may have more than one edge connecting them, with the adjacency matrix containing positive integers ($\mathbf{A} \in \mathbb{N}^{n \times n}$), as opposed to binary values. Figure 3.1 shows an example of a simple, undirected graph.

Graphs can be very powerful in formalizing and modeling real-world concepts. A relevant example for this chapter is that of a social network, where vertices are individuals and edges represent whether they are friends. There are a myriad of use-cases apart from this, in many different fields of study. This includes molecular graphs in chemistry to model molecules, connectomics in biology to model the nervous system, as well as various uses in computer science, such as network communication, computation flow, and a variety of discrete structures such as finite state automata.

In the field of machine learning, graph datasets are commonly used with special neural network architectures, graph neural networks, designed to take this data type as input for training and inference on a given task. Importantly, we can model our graph in a more elaborate manner by associating a feature vector with each individual node, or even edge. Going back to the social network example, the feature vectors associated with each node, i.e. individual in the social network, may consist of some information that describes that individual in more detail (e.g. age, gender, hobbies, and so forth). Depending on the task, a graph can have an associated label with each of its individual nodes, with the corresponding task referred to as node-level classification, or it can have a single label for the entire graph, for a graph-level classification task. In the latter case, information is aggregated over the network's final node representations for the final classification step. We describe one of the most well-known graph neural network architectures, the graph convolutional network (GCN) (Kipf and Welling, 2017), in more detail in Section 3.4.1.

3.2 Motivation and Contributions

Many text classification tasks naturally occur in the form of graphs where nodes represent text documents and edges are task specific, such as articles citing each other or health records belonging to the same patient. When learning node representations and predicting their categories, models benefit from exploiting information from the neighborhood of each node, as shown in graph neural networks, and graph convolutional networks in particular, making them superior to other models (Xu et al., 2019; De Cao et al., 2019).

While GCNs are powerful for a variety of NLP problems, like other neural models they are prone to privacy attacks. Adversaries with extensive background knowledge and computational power might reveal sensitive information about the training data from the model, such as reconstructing information about the original classes of a model (Hitaj et al., 2017) or even auditing membership of an individual's data in a model (Song and Shmatikov, 2019).

In order to preserve privacy for graph NLP data, models have to protect both the textual nodes and the graph structure, as both sources carry potentially sensitive information. As discussed in Chapter 2, privacy-preserving techniques, such as differential privacy, prevent information leaks by adding 'just enough' noise during model training while attaining acceptable performance, with the DP-SGD algorithm being the most prominent example in the application of machine learning. However, by design, DP-SGD expects independent and identically distributed (i.i.d.) data examples to form batches and 'lots', therefore its suitability for the case of graph neural networks remains an open question.

Within this chapter, we aim to tackle this exact problem, proposing a methodology for the application of DP-SGD to graph convolutional networks. Our goals are in line with the primary objectives of differential privacy to find a good privacy/utility trade-off for the resulting models, but importantly we take into account the privacy protection of the *entire* graph, as opposed to only node- or edge-related features. The main idea behind our method is to split the main graph from a particular dataset into subgraphs, while avoiding any additional queries on the data, i.e. the graph structure and its associated node or edge features.

Our contributions can be listed as follows:

1. We carry out experiments on several NLP text classification tasks, such as research article classification in citation networks, Reddit post classification, as well as user interest classification in social networks. We employ five different datasets in two languages, English and Slovak.

2. We demonstrate that it is possible to apply training with differential privacy to GCN models by using our graph splitting method, as well as proper optimization, which help the models to recover from performance drops due to the noisy setting of DP-SGD.

3. We further demonstrate that this performance drop can further be relieved by incorporating more sophisticated text representations. More specifically, with private training we are able to obtain a relative performance of 90% of the non-private setting, reaching a good privacy/utility trade-off at $\varepsilon = 1.0$.

4. To the best of our knowledge, the current study presented in this chapter is the first to bring differentially private gradient-based training to graph neural network models.

3.3 Related Work and Background

A wide range of NLP tasks have been utilizing **graph neural networks** (GNNs), specifically graph convolutional networks (GCNs), including text summarization (Xu et al., 2020a), machine translation (Marcheggiani et al., 2018), and semantic role labeling (Zheng and Kordjamshidi, 2020). Recent end-to-end approaches combine pre-trained transformer models with GNNs to learn graph representations for syntactic trees (Sachan et al., 2020). Rahimi et al. (2018) demonstrated the strength of GCNs on predicting geo-location of Twitter users where nodes are represented by users' tweets and edges by social connections, i.e. mentions of other Twitter users. However, for protecting user-level privacy, the overall social graph has to be taken into account.

As discussed in Chapter 2, gradient-based approaches to differential privacy (Williams and McSherry, 2010; Jain et al., 2012; Song et al., 2013; Bassily et al., 2014; Abadi et al., 2016b) pioneered the connection of DP and deep learning, with methods such as DP-SGD bounding the query sensitivity using gradient clipping and allowing for a lower total ε-DP guarantee over several epochs of training with the moments accountant. While originally tested on image recognition, they inspired subsequent work in language modeling using LSTMs (McMahan et al., 2018), as well as several other approaches (e.g. Hoory et al. (2021); Anil et al. (2022), see Section 2.2.5 of Chapter 2 for more discussion). However, to the best of our knowledge, training graph-based architectures with DP-SGD has not yet been explored, at the time the current chapter's investigation was carried out. Two recent approaches utilize *local differential privacy*, adding noise to each node before passing it to graph model training (Sajadmanesh and Gatica-Perez, 2020; Lyu et al., 2020b), yet it is unclear whether it prevents leaking knowledge about edges. Our setup is different as we have access to the full dataset and preserve privacy of the entire graph, including the adjacency matrix, representing information on all edge connections between nodes, as well as the node features themselves.

As GCNs typically treat the entire graph as a single training example, Chiang et al. (2019) proposed a more efficient training using mini-batching methods. Despite the NP-hardness of the general graph splitting problem (Bui and Jones, 1992), they experimented with random partitioning and other clustering methods that take advantage of the graph structure (Karypis and Kumar, 1998). It remains an open question whether splitting the graph into disjoint i.i.d. examples would positively affect our DP approach, where mini-batches/lots parameterize the required amount

of noise.

3.4 Methodology

3.4.1 GCN as the underlying architecture

We employ the GCN architecture (Kipf and Welling, 2017) for enabling DP in the domain of graph-based NLP. The GCN is a common and simpler variant to more complex types of GNNs, which allows us to focus our attention primarily on the application of DP to the domain of graph-based machine learning, without the potential interference of other additions to the base GCN architecture (e.g. use of edge features, inclusion of weight matrices for edges, other elements such as attention), providing a clear comparison of the DP and non-DP implementations of our model.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ model our graph data where each node $\mathbf{v}_i \in \mathcal{V}$ contains a feature vector of dimensionality c. The GCN aims to learn a node representation by integrating information from each node's neighborhood. The features of each neighboring node of $\mathbf{v}_i$ pass through a 'message passing function' (usually a transformation by a weight matrix $\mathbf{\Phi}$) and are then aggregated and combined with the current state of the node $\mathbf{h}_i^l$ to form the next state $\mathbf{h}_i^{l+1}$. Edges are represented using an adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, where n is the number of nodes in the graph. $\mathbf{A}$ is then multiplied by the matrix $\mathbf{H} \in \mathbb{R}^{n \times f}$, f being the hidden dimension, as well as the weight matrix $\mathbf{\Phi}$ responsible for message passing, learned during training. Additional tweaks by Kipf and Welling (2017) include adding the identity matrix to $\mathbf{A}$ to include self-loops in the computation $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, as well as normalizing matrix $\mathbf{A}$ by the degree matrix $\mathbf{D}$, specifically using a symmetric normalization $\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$. This results in the following equation for calculating the next state of the GCN for a given layer l, passing through a non-linearity function σ:

$$\mathbf{H}^{l+1} = \sigma(\hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{\Phi}^{(l)}) \tag{3.1}$$

The final layer states for each node are then used for node-level classification, given output labels. Figure 3.2 shows the general structure of a graph convolutional network.

3.4.2 Our approach: Graph cuts for improved DP performance

We propose a simple yet effective treatment of the discrepancy between GCN training (that is, taking the entire graph as a single example to maximally utilize the contextual information of each node) and the DP versions of SGD and Adam (DP-SGD and DP-Adam, respectively), which require a set of i.i.d. examples to form batches and lots in order to distribute DP noise effectively.

The unit for which our method provides a DP guarantee is a full graph, including all of its nodes and edges, which contrasts with other notions of DP for graphs such

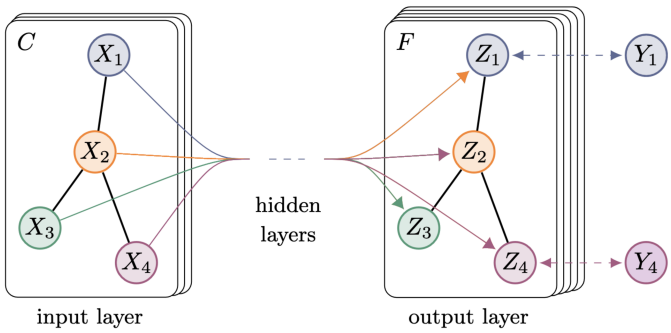


Figure 3.2: Diagram of a graph convolutional network. C represents the dimension of each input node, while F the dimension of each output node. Each node $\mathbf{X}_i$ goes through a series of hidden layers to obtain representations $\mathbf{Z}_i$. Y_i represent labels for each node. Figure taken from Kipf and Welling (2017).

as Edge DP and Node DP (Kasiviswanathan et al., 2013), which only protect edges, or nodes with all of their adjacent edges, respectively. As DP operates with the notion of neighboring datasets (Desfontaines and Pejó, 2020, Sec. 4), training a GCN privately on the full graph means that *any other graph* is neighboring. It also implies that each individual's privacy in that graph is protected, which is *the* goal of differential privacy. The other extreme would be to completely ignore the graph structure and train the GCN on individual nodes; using DP, it would again protect each individual's privacy, but any advantage of graph structure would be ignored.

We thus propose a sweet-spot approach, that is splitting the graph into disconnected subgraphs. We experiment with different numbers of subgraphs to find the best trade-off. In order to avoid any further dataset queries that might require a larger privacy budget, we utilize random masking of the adjacency matrix so no additional DP mechanism is required. Our algorithm creates a random index tensor for all nodes in the training set, which is then split into s groups, corresponding to the number of desired subgraphs. If the number of nodes n is divisible by s, then all subgraphs have equal sizes of nodes ($\frac{n}{s}$). If n is not divisible by s, then $n \bmod s$ subgraphs have $\lfloor \frac{n}{s+1} \rfloor$ nodes, while the rest have $\lfloor \frac{n}{s} \rfloor$ These indexes are then used to mask the original graph during training. This step is performed once during data preprocessing and requires very little additional computational time or memory requirements.

Privacy guarantee of graph cuts We start by summarizing the main DP argument of DP-SGD (Abadi et al., 2016b). In particular, any two gradient vectors are made 'indistinguishable' from each other up to factor $\exp(\varepsilon)$ and summand δ. Gradients are computed over mini-batches. This means that the presence or absence of an individual in the mini-batch is protected by DP-SGD. Furthermore, mini-batches are disjoint, such that each individual is associated with a unique mini-batch only. The disjoint requirement stems from DP being defined through the notion of neighboring datasets, where a given individual's record in the dataset cannot appear in any of its neighboring datasets (see Section 2.2). DP-SGD with lots and mini-batches is (ε, δ)-DP (Abadi et al., 2016b). In our graph scenario, we cut the graph into

several disconnected subgraphs that are equivalent to 'mini-batches'. When trained by DP-SGD, the gradients are again computed over each 'mini-batch' (subgraph) and privatized. Now having each individual (a single node and all its edges) in one 'mini-batch', the presence or absence of that individual is again protected by DP as desired. Therefore DP-SGD on GNNs with disjoint subgraphs is (ε, δ)-DP.

Finally, our graph splitting algorithm is completely random. It does not query (in the DP sense) any information about the graph and its output is independent of the presence or absence of any individual. As such, the random graph splitting algorithm does not consume any privacy budget. Overall, this makes our approach (ε, δ)-DP.

Our method is thus easy to implement, does not require much computational overhead and fits very well into the DP scenario. We extensively compare our results using different subgraph sizes in the non-private and private settings in Section 3.6.

3.5 Experiments

3.5.1 Datasets

We are interested in a text classification use-case where documents are connected via undirected edges, forming a graph. While structurally limiting, this definition covers a whole range of applications. We perform experiments on five single-label multi-class classification tasks. The **Cora**, **Citeseer**, and **PubMed** datasets (Yang et al., 2016; Sen et al., 2008; McCallum et al., 2000; Giles et al., 1998) are widely used citation networks of research papers, where citing a paper i from paper j creates an edge $i - j$. The task is to predict the category of the particular paper.

The **Reddit** dataset (Hamilton et al., 2017) treats an original post, i.e. content shared in a given Reddit community known as a subreddit, as a graph node and connects two posts by an edge if any user commented on both posts. Given the large size of this dataset (230k nodes; all posts from Sept. 2014) causing severe computational challenges, we sub-sampled 10% of posts (only a few days of Sept. 2014). The gold label corresponds to one of the top Reddit communities to which the post belongs.

Unlike the above English datasets, the **Pokec** dataset (Takac and Zabovsky, 2012; Leskovec and Krevl, 2014) contains an anonymized social network in Slovak. Nodes represent users and edges their friendship relations. User-level information contains many attributes in natural language (e.g. 'music', 'perfect evening'). We set up the following binary task: Given the textual attributes, predict whether a user prefers dogs or cats. We decided against user profiling, namely age prediction for ad targeting (Perozzi and Skiena, 2015), for ethical reasons. Our task still serves well the demonstration purposes of text classification of social network data. Overall, Pokec's personal information including friendship connections shows the importance of privacy-preserving methods to protect this potentially sensitive information. We discuss further preparation details below in Section 3.5.2.

The four English datasets adapted from previous work are only available in their encoded form. For the citation networks, each document is represented by a Bag of Words (BoWs) encoding.

The Reddit dataset combines GloVe vectors (Pennington et al., 2014) averaged over the post and its comments. Only the Pokec dataset is available as raw texts, so we opted for multilingual BERT (mBERT) (Devlin et al., 2019), and averaged all contextualized word embeddings over each user's textual attributes. We additionally tried Sentence-BERT (Reimers and Gurevych, 2019), which resulted in lower performance. We hypothesize this is due to the fact that users fill in each of the attributes in such a way that the text tends to resemble a list of keywords, rather than actual discourse. Overall, the variety of languages, sizes, and different input encoding allows us to compare non-private and private GCNs under different conditions. Table 3.1 summarizes the data sizes and number of classes.

Dataset	Classes	Test size	Training size
CiteSeer	6	1,000	1,827
Cora	7	1,000	1,208
PubMed	3	1,000	18,217
Pokec	2	2,000	16,000
Reddit	41	5,643	15,252

Table 3.1: Dataset statistics; size is number of nodes.

3.5.2 Further details on Pokec dataset pre-processing

In order to prepare the binary classification task for the Pokec dataset, the original graph consisting of $1,632,803$ nodes and $30,622,564$ edges is sub-sampled to only include users that filled out the 'pets' column and had either cats or dogs as their preference, discarding entries with multiple preferences. For each pet type, users were reordered based on percent completion of their profiles, such that users with most of their profile information present were retained.

For each of the two classes, the top $10,000$ users are taken, with the final graph consisting of $20,000$ nodes and $32,782$ edges. The data was split into 80% training, 10% validation and 10% test partitions.

The textual representations were prepared with `bert-base-multilingual-cased` from Huggingface transformers,[1] converting each attribute of user input in Slovak to mBERT embeddings with the provided tokenizer for the same model. Embeddings are taken from the last hidden layer of the model, with dimension size 768. The average over all tokens is taken for a given column of user information, with 49 out of the 59 original columns retained. The remaining 10 are left out due to containing less relevant information for textual analysis, such as a user's last login time. To further simplify input representations for the model, the average is taken over all

[1] Available from https://huggingface.co/bert-base-multilingual-cased.

columns for a user, resulting in a final vector representation of dimension 768 for each node in the graph.

3.5.3 Experiment Setup

We operate with three benchmarking scenarios. **Experiment A** is the vanilla GCN without DP: The aim is to train the GCN without any privacy mechanism, evaluating also the influence on performance with less training data. **Experiment B** is the GCN with DP: We evaluate performance varying the amount of privacy budget as well as data size. We randomly sub-sample a certain percentage of nodes that then form a single training graph, as in the standard GCN. The latter allows us to see the effects on performance of both adding noise and reducing training data. **Experiment C** is the GCN with graph splits: Evaluating performance varying the number of graph splits in the non-DP and DP settings.

Implementation details. As the δ privacy parameter is typically kept 'cryptographically small' (Dwork and Roth, 2013) and, unlike the main privacy budget ε, has a limited impact on accuracy (Abadi et al., 2016b, Fig. 4), we fixed its value to 10^{-5} for all experiments. The clipping threshold C is set at 1.0. We validated our PyTorch (Paszke et al., 2019) implementation by fully reproducing the MNIST results from (Abadi et al., 2016b), described in Section 3.6.6. We perform all experiments five times with different random seeds and report the mean and standard deviation. Early stopping is determined using the validation set. We discuss further details regarding hyperparameters below.

3.5.4 Hyperparameter Configuration

Our GCN model consists of 2 layers, with a ReLU non-linearity, a hidden size of 32 and dropout of 50%, trained with a learning rate of 0.01 (apart from DP-Adam, which required far higher learning rates, as mentioned below). We found that early stopping the model works better for the non-DP implementations, where we used a patience of 20 epochs. We did not use early stopping for the DP configuration, which shows better results without it. For all SGD runs we used a maximum of 2000 epochs, while for Adam we used 500.

Importantly, for DP-Adam we noticed that more moderate learning rate values such as 0.01 were insufficient and led to far lower performance. We therefore optimized this at several values in the interval from 0.1 to 100, with some datasets and graph split values requiring learning rates as low as 0.1 (e.g. most datasets with 100 graph splits), while in other cases requiring 50 or 100 (e.g. Reddit for most graph split values).

Due to the smaller amount of epochs for Adam, it is possible to add less noise to achieve a lower ε value. Table 3.2 shows the mapping from noise values used for each optimizer to the corresponding ε in the full graph setting. We implement the DP-SGD and the corresponding DP-Adam algorithms from scratch using the PyTorch library (Paszke et al., 2019), with the exception of the accounting procedure, for which we use the provided command-line tool from the TensorFlow Privacy library

(Abadi et al., 2016a).[2] The runtimes for our experiments reach up to 1 hour for the larger configurations on an NVIDIA A100 Tensor Core GPU.

ε	**Noise-SGD**	**Noise-Adam**
136.51	4	2
9.75	26	13
4.91	48	24
2.00	112	56

Table 3.2: ε values from **Experiment B**, with the corresponding noise values added to the gradient for each optimizer.

Finally, regarding hyperparameter optimization on the validation set in the DP setting, Abadi et al. (2016b) mention that, when optimizing on a very high number of parameter settings (e.g. in the thousands), this would additionally take up a moderate privacy budget (e.g. $\varepsilon = 4$ if they had used $6,700$ hyperparameters). For our experiments, this number of hyperparameters is comparatively minimal and would be well within our privacy bounds.

3.6 Results and Analysis

Non-DP			**DP**			**DP split**	
Maj.	SGD	Adam	ε	SGD	Adam	SGD	Adam
CiteSeer			1	-	-	0.35	0.36
0.18	0.77	0.79	2	**0.36**	**0.36**	0.35	**0.36**
Cora			1	-	-	0.55	0.56
0.32	0.77	0.88	2	0.39	0.52	0.55	**0.57**
PubMed			1	-	-	0.54	0.52
0.40	0.49	0.79	2	0.38	**0.54**	**0.54**	0.51
Pokec			1	-	-	0.62	0.72
0.50	0.83	0.83	2	**0.75**	0.66	0.64	0.73
Reddit			1	-	-	0.65	0.79
0.15	0.68	0.88	2	0.46	0.72	0.67	**0.82**

Table 3.3: F_1 results for **Experiments A, B and C**: Full dataset without DP (first three columns, including a majority baseline), with DP and varying ε (middle two columns), with DP using graph splits (right-most two columns). Best DP results are bold. Lower ε corresponds to better privacy.

3.6.1 Experiment A: Non-private GCN

Table 3.3 shows the results on the left-hand side under 'Non-DP'. When trained with SGD, all datasets achieve fairly good results with the exception of PubMed,

[2] Available at https://github.com/tensorflow/privacy/blob/master/tensorflow_privacy/privacy/analysis/compute_dp_sgd_privacy.py.

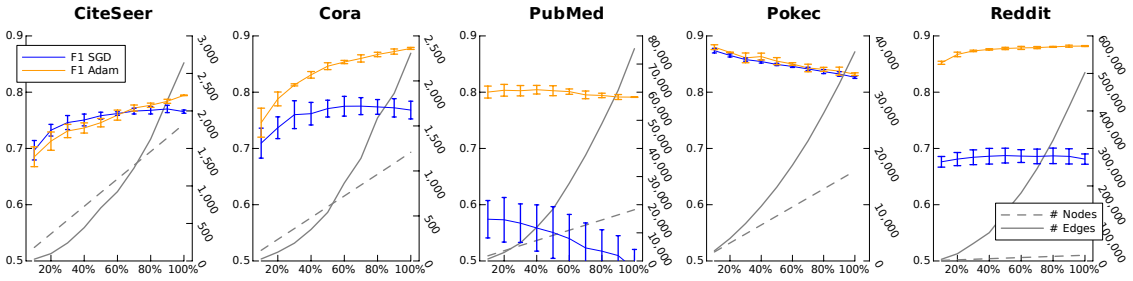


Figure 3.3: **Experiment A**: F_1 wrt. training data size (in %), without DP.

possibly due to PubMed having a much larger graph. The best of these is for Pokec, which could be due to its more expressive representations (BERT) and a simpler task (binary classification). In comparison, in line with previous research (Ruder, 2016), Adam outperforms SGD in all cases, with Pokec showing the smallest gap (0.826 and 0.832 for SGD and Adam, respectively).

Figure 3.3 shows the non-DP results with increasing training data. We observe two contrasting patterns. First, there is a clear improvement as training data increases (e.g. CiteSeer, with 0.70 F1 score at 10% vs. 0.77 at 100%). Second, we observe the exact opposite pattern, with PubMed dropping from 0.57 at 10% to 0.49 at 100%, with a similar pattern for Pokec, or an early saturation effect for Reddit and Cora, where results do not increase beyond a certain point (at 20-30% for Reddit with approximately 0.69 F1 score, 50% for Cora at a score of 0.77). We speculate that, with a larger training size, a vanilla GCN has a harder time to learn the more complex input representations. In particular, for PubMed and Pokec, the increasing number of training nodes only partially increases the graph degree, shown in the solid and dotted grey lines of Figure 3.3, respectively. The model thus fails to learn expressive node representations when limited information from the node's neighborhood is available. In contrast, the graph degree of Reddit grows much faster, thus advantaging GCNs.

3.6.2 Experiment B: GCN with DP

The middle columns of Table 3.3 show results for a privacy budget of $\varepsilon = 2.0$. As discussed further below, without splitting the graph into subgraphs, it is impossible to reach the lower ε value of 1.0, since computations become very unstable due to the very large amount of noise. We do not report values larger than $\varepsilon = 2.0$ as their privacy protection diminishes exponentially. We note four main patterns in this experiment.

First, **DP-SGD results stay the same, regardless of the noise value added.** This is quite unexpected, since higher added noise values would be anticipated to lead to lower results. One explanation for this pattern is that the gradients in vanilla SGD are already quite noisy, which may even help in generalization for the model, so the additional DP noise does not pose much difficulty beyond the initial drop in performance.

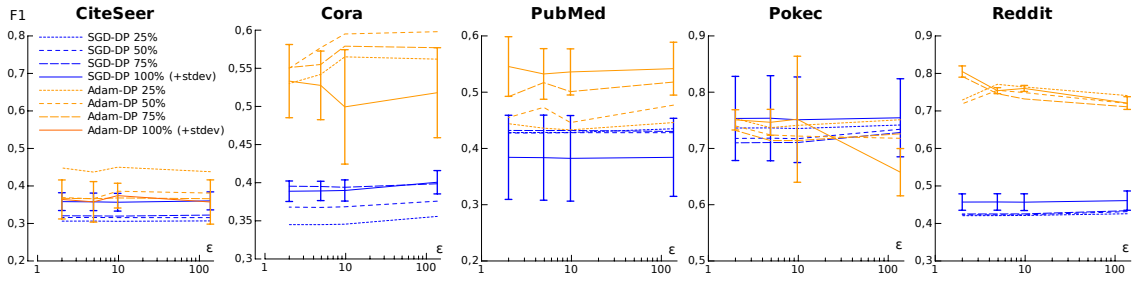


Figure 3.4: **Experiment B**: F_1 with varying training data size (in %) wrt. privacy budget ε, with DP.

Second, **DP-Adam results outperform DP-SGD and can reach results close to the non-DP settings.** It is worth noting that, when using default hyperparameters with a moderate learning rate of 0.01, DP-Adam results are very low, usually worse than DP-SGD. It is only when optimizing this learning rate that we see substantial improvements, with the best-performing learning rates being very large, in the case of Reddit as high as 100. In contrast, DP-SGD does not see much benefit from additional hyperparameter optimization.

Third, we see **bigger drops in performance in the DP setting for datasets with simpler input representations.** Datasets of simpler input features can have results drop by more than half in comparison to the non-DP implementation. In comparison to DP-SGD, DP-Adam is able to retain better performance even with the simpler input features for Cora and PubMed (e.g. drop 0.79 → 0.54 for PubMed). Reddit and Pokec show the smallest drops from the non-DP to DP setting (0.88 → 0.72 and 0.83 → 0.66 with Adam, for each dataset, respectively). In fact, even for DP-SGD, Pokec results are very close to the non-DP counterpart (0.83 → 0.75 F1 score). Hence, Citeseer, Cora and PubMed, all using one-hot textual representations, show far greater drops in performance for the DP setting. The datasets utilizing GloVe (Reddit) and BERT (Pokec) representations perform far better. Since this effect of feature complexity on DP performance is shown only through different datasets, we perform additional experiments on Pokec using BoWs, fastText (Grave et al., 2018) and BERT features for a proper 'apples-to-apples' comparison described in Section 3.6.4.

Learning Curves with DP Figure 3.4 shows the DP results for both varying ε and with different training sub-samples (25%, 50%, 75% and the full 100%). First, generally observed patterns are not the same for the learning curves in the non-DP setup (Experiment A). For instance, Adam exhibits the opposite pattern, e.g. Citeseer and Cora increase with more data for Adam without DP, but decrease for DP-Adam.

Second, we can see that **increasing the amount of data does not necessarily help in the DP setting**. For instance, while there is an improvement for DP-SGD with the Citeseer, Cora and Reddit datasets, results mostly get worse for DP-Adam, with the exception of PubMed. Hence, increasing training data generally does not act as a solution to the general drop in performance introduced by DP.

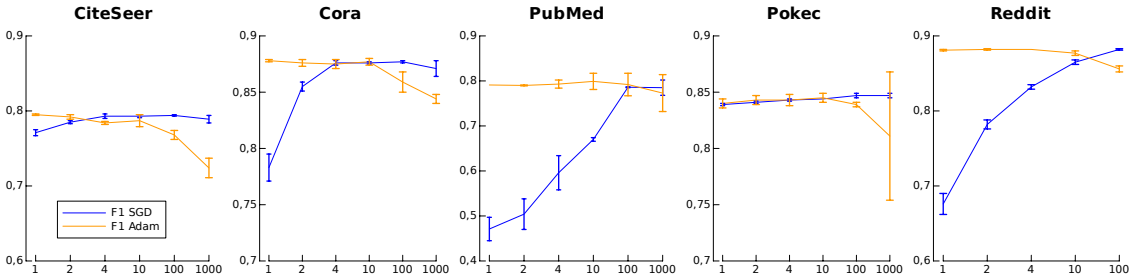


Figure 3.5: **Experiment C, no DP**: F_1 wrt. number of subgraphs.

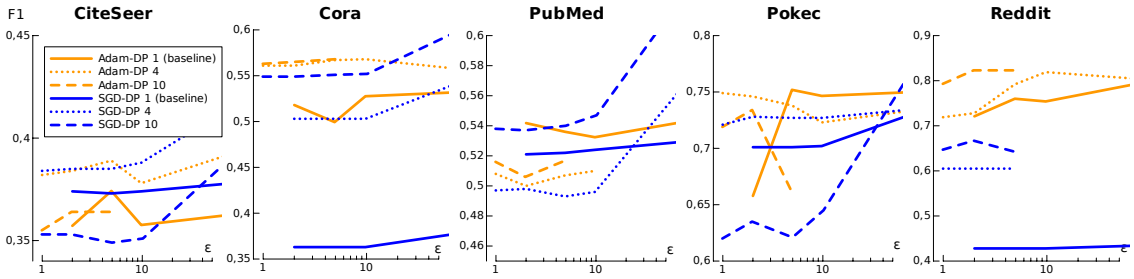


Figure 3.6: **Experiment C, with DP**: F_1 with varying number of subgraphs wrt. privacy budget ε.

3.6.3 Experiment C: Graph splitting approach

First we highlight the main results for the **non-DP graph splitting** approach. For all datasets, increasing the number of subgraph divisions yields better results in the SGD setting. This is especially notable in a case such as for PubMed, where there is an increase from 0.47 with no splits to 0.79 with a split size of 100. Overall, splitting the graph is shown to be quite effective in a setting where the model may struggle more in the learning process, such as with SGD. Furthermore, at the higher subgraph split sizes, there is a slight drop-off for Adam but not for SGD, where increasing subgraph split size never improves performance beyond vanilla Adam, as shown in Figure 3.5.

Figure 3.6 shows the results for the **graph splitting setting with DP**, varying the privacy budget. First, **increasing the number of subgraph splits generally improves results** for DP-SGD (e.g. 0.36 $\rightarrow$ 0.55 for Cora at $\varepsilon = 2.0$, with 10 subgraphs vs. the full graph, respectively). We see a difference across datasets, where for instance Reddit shows the best results at 10 splits for all three ε values, while Citeseer or Pokec do not particularly benefit beyond four splits.

Second, this pattern of improvement is also noticeable in the case of DP-Adam, in contrast to the non-private vanilla Adam results. This is clearly seen in the Reddit results, where the very best result is with 10 splits with $\varepsilon = 1.0$ at an F-score of 0.79 with DP-Adam, being just 0.09 points lower than the non-DP version. As in Experiment B, it is notable that DP-Adam does not perform particularly well without using very high learning rates, with less graph splits requiring larger learning rate values. Overall, the best-performing number of subgraphs seems to be specific

to the particular dataset used and can thus be treated as another hyperparameter to optimize on.

Third, with more subgraphs allowing for less noise to be added to obtain a lower ε value, it is possible to **reach the strong privacy guarantee** of $\varepsilon = 1.0$. For comparison, the randomized response technique described in Section 2.2.3 (Warner, 1965), using fair coin flips with uniform outcome probabilities, has $\varepsilon \approx 1.1$. Thus our graph splitting approach not only helps mitigate the difficulties of training with added DP noise, but also allows us to reach stronger privacy guarantees with about the same performance. Without the mini-batching that becomes possible by splitting the graph, it is impossible to carry out a stable computation during the moment accounting process to achieve $\varepsilon = 1.0$. A comparison of the DP setting with graph splitting (using our initial setup of 10 graph splits) and the regular DP setting is summarized in Table 3.3.

Summary and take-aways We summarize the key observations as follows:

1. DP-SGD is fairly robust to noise for these datasets and settings, even at $\varepsilon = 1.0$.

2. DP-Adam works even better than DP-SGD, however it needs to be tuned very carefully, using very high learning rates.

3. More complex representations are better for the DP setting, showing a smaller performance drop from the non-DP results.

4. Increasing training data does not necessarily mitigate negative performance effects of DP.

5. Graph splitting improves both performance and allows for a stronger privacy guarantee of $\varepsilon = 1.0$, resolving the mini-batching problem of GCNs in the DP setting.

In addition to the above set of main experiments, we perform additional experiments to elaborate on some of our findings above. First, we investigate observation (3) on representation complexity in Section 3.6.4, to have an 'apples-to-apples' comparison of different textual representations on a single dataset (Pokec). Next, we perform an error analysis on 'hard cases' in the non-DP and DP settings in Section 3.6.5, to look into whether the most difficult data points for the model are common between the various settings of experiments A and B, i.e. low-data and high DP noise. Finally, we reproduce the results of Abadi et al. (2016b) on the MNIST dataset in Section 3.6.6.

3.6.4 Feature Comparison for Pokec Dataset

As mentioned in Sections 3.6.2 and 3.6.3, we perform additional experiments on the Pokec dataset in order to further investigate the hypothesis that input feature complexity has an effect on the degree of performance drop in the DP setting. We

originally noticed this across separate datasets, with Cora, Citeseer and PubMed, using one-hot input features, having a greater performance drop than Reddit and Pokec, which use GloVe and BERT, respectively. In order to more properly evaluate this under the same conditions, we prepare two additional types of input features for the Pokec dataset, namely Bag of Words (BoWs) and fastText (Joulin et al., 2017), altogether having three levels of word representations, ranging from the simpler (BoWs) to more complex (BERT).

The BoWs embeddings were prepared by taking the same 20,000 user profiles as in the BERT preprocessing methodology described above. Tokens were split on whitespace, with additional steps such as punctuation removal and lowercasing. In order to reduce the embedding dimensionality, we filtered tokens by frequency in the interval $[15, 15000]$. Each user profile was thus represented with a 9447-dimensional vector of binary values.

For the fastText embeddings, we use a pre-trained model for Slovak from Grave et al. (2018). Using the same set of user profiles, we preprocess the data in the same manner as described by the authors. In order to have one vector per user profile, we average all fastText embeddings for a given user to have a final embedding dimension of 300.

The results of this experiment can be seen in Table 3.4. We notice that, in line with our hypothesis, the most effective input features in the DP setting are the BERT embeddings, with the smallest performance drop from the non-DP setting (e.g. $0.84 > 0.70$ for DP-SGD at $\varepsilon = 2.0$). Interestingly, the best method overall without DP is with the BoWs representation. One explanation for this is that a lot of slang vocabulary and unusual tokens are used in the social network data, which a fastText or BERT model may struggle with more, while BoWs would simply treat them equally as any other token in the vocabulary. As expected, the BoWs embeddings have a larger drop when trained with DP ($0.88 > 0.62$ for SGD and DP-SGD with $\varepsilon = 2.0$, respectively). The fastText results show the lowest performance both in the non-DP and DP settings, possibly due to the model struggling to maintain useful representations after averaging many token vectors for a user, which a more powerful model such as mBERT has an easier time with. Our original hypothesis is thus verified that more sophisticated input features such as BERT would show a smaller performance drop in the DP setting, compared to simpler representations such as BoWs, with this effect shown in an 'apples-to-apples' setting on the same dataset.

3.6.5 Are 'hard' examples consistent between private and non-private models?

To look further into the nature of errors for experiments A and B, we evaluate the 'hard' cases. These are cases that the model has an incorrect prediction for with the maximum data size and non-private implementation, i.e. the first set of results from experiment A. For the experiment A learning curves, we take the errors for every setting of the experiment (10% training data, 20%, and so forth) and calculate the

Non-DP F_1 scores		**DP** F_1 scores		
SGD	Adam	ε	SGD	Adam
BoWs		2	0.62	0.63
0.88	0.87	5	0.62	0.61
		10	0.62	0.61
		137	0.63	0.63
fastText		2	0.59	0.63
0.71	0.73	5	0.59	0.57
		10	0.59	0.57
		137	0.61	0.60
BERT		2	0.70	0.66
0.84	0.84	5	0.70	0.75
		10	0.70	0.75
		137	0.74	0.75

Table 3.4: F_1 scores for the Pokec dataset, comparing different input feature representations and privacy budgets.

intersection of those errors with that of the 'hard cases' from the baseline implementation. This intersection is then normalized by the original number of hard cases to obtain a percentage value. The results for the hard cases of experiment A can be seen in Figure 3.7. We perform the same procedure for the hard cases of experiment B with different noise values for the DP-SGD setting, as seen in Figure 3.8. This provides a look into how the nature of errors differs among these different settings, whether they stay constant or become more random as we decrease the training size or increase DP noise.

Regarding the errors for experiment B, we can see a strong contrast between datasets such as Reddit and PubMed. For the latter, the more noise we add as ε decreases, the more random the errors become. In the case of Reddit, however, we see that even if we add more noise, it still fails on the same hard cases. This means that there are hard aspects of the data that remain constant throughout. For instance, out of all the different classes to predict, some may be particularly difficult for the model.

Although the raw data for Reddit does not have references to the original class names and input texts, we can still take a look into these classes numerically and see which ones are the most difficult in the confusion matrix. In the baseline non-DP model, we notice that many classes are consistently predicted incorrectly. For example, class 10 is predicted 93% of the time to be class 39. Class 18 is never predicted to be correct, but 95% of the time predicted to be class 9. Class 21 is predicted as class 16 83% of the time, and so forth. This model therefore mixes up many of these classes with considerable confidence.

Comparing this with the confusion matrix for the differentially private implementation at an ε value of 2, we can see that the results incorrectly predict these same classes as well, but the predictions are more spread out. Whereas the non-

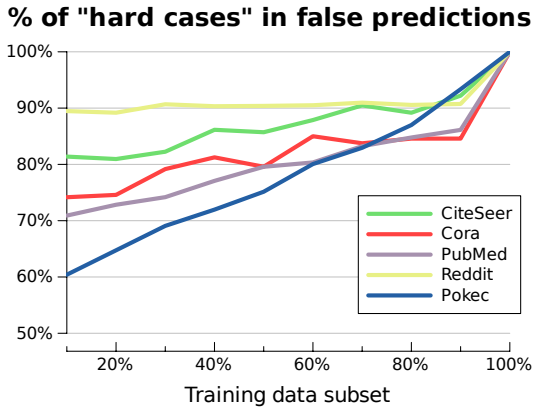


Figure 3.7: Hard cases in the non-DP setting.

private model seems to be very certain in its incorrect prediction, mistaking one class for another, the private model is less certain and predicts a variety of incorrect classes for the target class.

For the analysis of the hard cases of experiment A in Figure 3.7, we can see some of the same patterns as above, for instance between PubMed and Reddit. Even if the training size is decreased, the model trained on Reddit still makes the same types of errors throughout. In contrast, as training size is decreased for PubMed, the model makes more and more random errors. The main difference between the hard cases of the two experiments is that, apart from Reddit, here we can see that for all other datasets the errors become more random as we decrease training size. For example, Cora goes down from 85% of hard cases at 90% training data to 74% at 10% training data. In the case of experiment B, they stay about the same. For instance, Cora retains just over 70% of the hard cases for all noise values.

Overall, while we see some parallels between the hard cases for experiments A and B, with respect to patterns of individual datasets such as Reddit and PubMed, the general trend of more and more distinct errors that is seen for the majority of datasets with less training size in experiment A is not the same in experiment B, staying mostly constant across different noise values for the latter. The idea that the nature of errors due to DP noise and the nature of errors due to having less training data are the same can thus not be confirmed. This means that simply increasing training size may not necessarily mitigate the negative performance effects of DP noise.

3.6.6 MNIST Baselines

Table 3.5 shows results on the MNIST dataset (LeCun et al., 1998) with different lot sizes and noise values. We use a simple feedforward neural network with a hidden size of 512, dropout of 50%, SGD optimizer, and a maximum of 2000 epochs with an early stopping patience of 20. Other hyperparameters such as learning rate are the same as described above. We note that the configuration in the first row with lot size of 600 and noise 4 is the same as described by Abadi et al. (2016b) in their

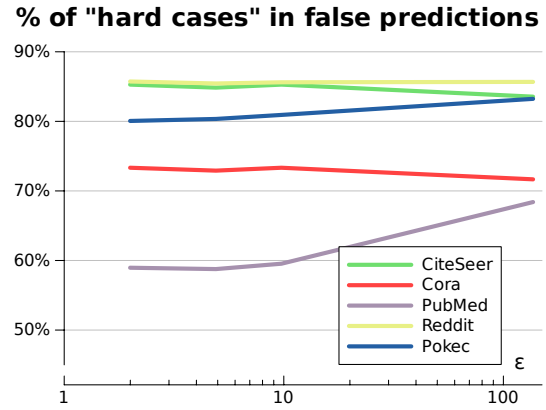


Figure 3.8: Hard cases analysis in the DP setting.

Lot Size	Noise	ε	F_1	Std.
600	4	1.26	0.90	0.02
6,000	4	4.24	0.84	0.01
60,000	4	15.13	0.45	0.04
60,000	50	0.98	0.39	0.15
60,000	100	0.50	0.10	0.01

Table 3.5: Results on the MNIST dataset with varying lot sizes and noise values.

application of the moments accountant, reaching the same ε value of 1.2586.

We can see some important patterns in these results that relate to our main results from the GCN experiments. Maintaining a constant noise scale of 4, as we increase the lot size, not only does the ε value increase, but we see a dramatic drop in F_1 score, especially for a lot size of $60,000$, being the full training set. If we try to increase the noise and maintain that $60,000$ lot size, while we are able to lower the ε value below 1, the F_1 score continues to drop dramatically, going down to 0.1010 with a noise value of 100.

Hence, the current MNIST results further show the benefits of applying the graph splitting methodology on large one-graph datasets. By splitting the graph, we are able to utilize batches and lots of smaller sizes, with smaller amounts of the input data processed per iteration, as opposed to running the entire dataset through the model each time.

3.7 Chapter Summary

We have investigated privatization strategies for neural models operating on graph datasets, i.e. graph neural networks. While there is an expected drop in results for the differentially private models, we can mitigate this drop by: (1) using graph partitioning methods, such as our random graph splitting methodology, (2) utilizing the DP-Adam optimizer, which interestingly requires very high learning rates that are unusual in standard hyperparameter optimization of neural models, and

(3) having more complexity in the input representations of the dataset (e.g. node features as mBERT embeddings v.s. BoWs). Our final privacy/utility trade-off is quite effective, reaching the strong privacy guarantee of $\varepsilon = 1.0$, with up to 87% and 90% of non-private F_1 scores for the Pokec and Reddit datasets, respectively.

With regards to further options in this direction of gradient-based GNN privatization, an interesting line of work to explore is graph splitting that may take advantage of the graph structure, instead of using uniform random splitting. The main issue is that any additional queries on the dataset require the incorporation of more noise, in order to maintain the (ε, δ)-DP guarantee. This is what made the random graph splitting approach appealing, avoiding these additional queries, and therefore additional perturbation on the graph dataset. Perhaps a good trade-off can be found, with a graph splitting approach that shows significant performance improvements, requiring only a small amount of perturbation to be privatized.

Having investigated strategies for privatizing graph neural network architectures in NLP, we next ask the question: How does this methodology of DP-SGD perform within the field of NLP in general, for 'standard' NLP datasets, tasks, and model types? Without the added difficulties that are present for graph neural networks and graph datasets, can we simply apply DP-SGD to models such as LSTMs and Transformers, and expect to reach a good privacy/utility trade-off? We thoroughly investigate this question in the following chapter.

Chapter 4

Investigating Strategies for Differentially-Private Learning across NLP Tasks

We continue our investigation of privatizing NLP models in the framework of differential privacy. In this chapter, we address RQ2 from Chapter 1:

RQ2 Is there a systematic strategy that can be applied to NLP text classification tasks in the differentially private setting?

Unlike in the previous chapter, here we investigate more 'standard' NLP architectures such as LSTM models (Hochreiter and Schmidhuber, 1997) and several transformer-based configurations (Vaswani et al., 2017), across a variety of NLP tasks. While differential privacy has been used in some NLP studies, the community overall does not have a full understanding of how well DP can be utilized across different NLP models and tasks.

4.1 Introduction

As we have previously discussed, one of the most standard ways of applying differential privacy to the field of machine learning is by using the DP-SGD algorithm. While this is becoming more widely adopted for private machine learning in general (Papernot et al., 2021; Ziller et al., 2021; De et al., 2022), within the NLP community we do not yet have a thorough understanding of how effective DP-SGD is in various NLP tasks. Going through the past literature on private NLP, DP-SGD has been used in some cases, such as for the task of language modeling (McMahan et al., 2018; Hoory et al., 2021), as well as named entity recognition (NER) (Jana and Biemann, 2021), as we describe in Section 2.2.5 of Chapter 2. In the latter case, some of the findings seem to be counter-intuitive, for instance a lack of decrease in performance when using very strict privacy guarantees. Overall, the current landscape of applying gradient-based differential privacy to the field of NLP lacks some

concrete answers on the effectiveness of the methodology, as well as what strategies may be employed in order to obtain both good performance and a stricter privacy guarantee. For instance, do certain model configurations tend to have a 'winning' combination with respect to the architecture that is used with DP-SGD? Are certain NLP tasks more suited for this methodology, with the possibility of achieving a better privacy/utility trade-off?

In this chapter we answer the following research questions. (1) We want to know which NLP models and strategies for training these models can provide the best trade-off with respect to privacy and performance across different text classification tasks. (2) If we increase the strictness of our privacy guarantees, to what extent does this hurt the final performance? (3) Is there a systematic strategy that can be employed with respect to (1) in order to obtain the best results, achieving the strictest possible degree of privacy and highest performance? We investigate these main points by running experiments over seven different NLP datasets and five NLP tasks. We compare different models that are commonly used in the current sphere of NLP research. The primary contribution of this chapter is to provide the NLP community with a more solid grasp of how privacy-preserving learning relates to each task and its associated challenges.

4.2 Related Work and Background

In the field of NLP, the primary application of DP-SGD has been for training language models. For instance, Kerrigan et al. (2020) look into the application of DP-SGD on a GPT-2 model (Radford et al., 2019) and two simple feedforward neural networks. They pre-train on a large public dataset and perform fine-tuning with differential privacy, reporting model perplexities on these pre-trained models. They do not perform further experiments on downstream tasks.

In McMahan et al. (2018), the authors prepare an LSTM language model trained with differential privacy. They are able to achieve results that are close to the non-private model counterparts. Next, Hoory et al. (2021) prepare a BERT model with differential privacy. They run experiments on a medical entity extraction task and using a privacy budget of $\varepsilon = 1.1$, they reach performance that is comparable between the DP and non-DP models.

With regards to downstream tasks in NLP, there are only a few works that utilize DP-SGD. Jana and Biemann (2021) investigate the CoNLL-2003 English NER dataset (Tjong Kim Sang and De Meulder, 2003) with differentially private training. Using DP-SGD with values of ε that go as low as 1, and even 0.022, they report no significant drop in performance, using a bidirectional LSTM. We evaluate this highly unusual result in more detail in Section 4.5.3 below.

In addition, Bagdasaryan et al. (2019) look into the NLP task of sentiment analysis for Tweets with DP-SGD. They utilize ε values of 8.99 and 3.87, achieving only small drops in accuracy.

The current state of the art is therefore limited to only a few tasks and datasets, with distinct privacy budgets that do not match up across these different investigations. As a community, we therefore need a more general investigation of the application of DP-SGD to the domain of NLP, exploring whether a common strategy can be found for this application across different tasks and models. We refer to Section 2.2.5 of Chapter 2 for a broader background into differentially private research in NLP.

Finally, in a concurrent work to the investigation presented in this chapter, Yu et al. (2022) examined pre-training and fine-tuning of transformer-based language models on a variety of tasks and using several parameter-efficient fine-tuning methods. They report higher results using larger models, with the parameter-efficient techniques helping to reduce added computation and memory costs.

4.2.1 What is being privatized

As is standard for the DP-SGD algorithm, the *unit of privacy* for our DP guarantee is that of individual data points in a dataset. In other words, the notion of a neighboring dataset for our experiments is based on adding or removing one data point (i.e. document) in each of the corresponding datasets that we utilize. This means that the contribution of every single document to the resulting model parameter update at each training iteration is made indistinguishable, up to a factor of ε and summand δ. Unlike in a database of individuals and corresponding traits, as described in Section 2.2.2 of Chapter 2, the contribution of each individual in a natural language dataset is not always straightforward to determine and requires further investigation. While we are working with DP at the stricter document level, obscuring the contribution of all tokens in the sequence, it may be possible to relax this requirement by operating at a more fine-grained level, depending on whether some tokens are considered more sensitive than others in a document, with respect to identifying the individual contributing the document (e.g. Shi et al. (2022a); Wu et al. (2022)). For related discussion, see Section 6.6.4 of Chapter 6.

4.3 Knowledge Distillation

Before delving into the exact models and tasks that are investigated throughout this chapter, we briefly discuss **knowledge distillation**, relevant to one class of models that will be used in our experiments, namely the XtremeDistilTransformer (XDT) model (Mukherjee et al., 2021). Knowledge distillation can be defined as the process of taking a larger model and transferring its knowledge to a smaller one (Ba and Caruana, 2014; Hinton et al., 2015). Typically, this larger model can be computationally expensive to deploy, with significant memory requirements, while the smaller model is much more feasible in practical, real-world settings. The procedure is common on large neural network architectures, especially the modern transformer-based models, consisting of a large number of model parameters and layers. Figure 4.1 shows the typical knowledge distillation pipeline.

The core elements of knowledge distillation include: (a) the *knowledge* itself, (b)

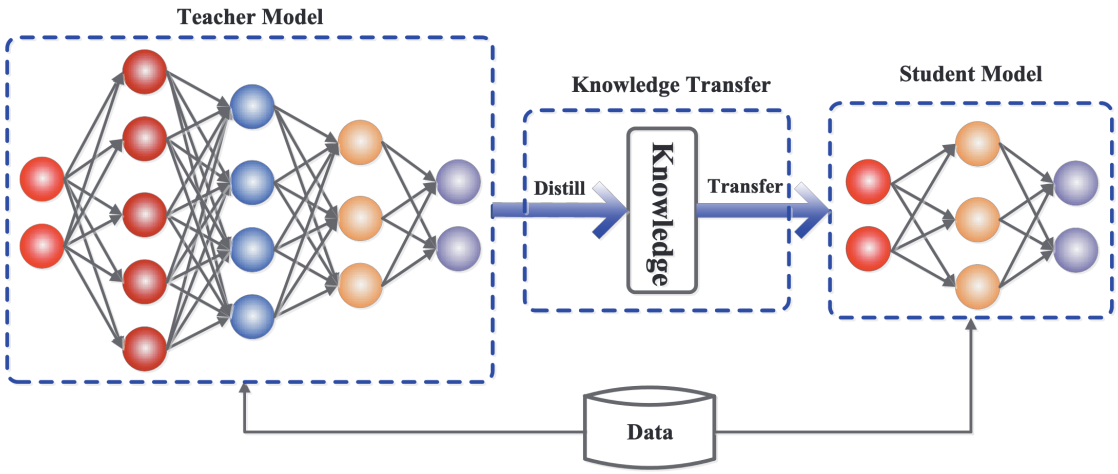


Figure 4.1: General pipeline for knowledge distillation, with a larger *teacher model* (left) transferring its learned representations to a smaller and more compact *student model* (right). Figure taken from Gou et al. (2021).

the *method*, or *type* of distillation, (c) the *teacher-student architecture*, as well as (d) the exact *algorithm* used for distillation (Gou et al., 2021). Each of these is outlined in more detail below.

Knowledge *Knowledge* can be grouped into several categories, primarily relating to the various learned elements of the larger model. These include *response-based knowledge* (Ba and Caruana, 2014; Hinton et al., 2015), i.e. information from the last output layer in the form of the teacher model's logits, *feature-based knowledge* (Romero et al., 2015), in which the student model learns from the intermediate layers of the teacher model (e.g. learning the same feature activations), as well as *relation-based knowledge* (Yim et al., 2017), in which the relationship between the various feature maps (e.g. two layers) of the teacher model are used to train the student model.

Method of Distillation There are three main ways of preparing a distilled student model: *offline* distillation, *online* distillation, and *self* distillation. The first of these, *offline* distillation (Hinton et al., 2015), takes a pre-trained model as the teacher, and transfers its knowledge to a student model (e.g. in the form of logits, intermediate knowledge, or pairs of feature maps, as above). This is the most common methodology out of the three methods of distillation. *Online* distillation (Zhang et al., 2018) has both the teacher and student models trained simultaneously. Thus, instead of the above two-step approach of offline distillation, here the whole procedure works end-to-end. Finally, in *self* distillation (Zhang et al., 2019a), the same model acts as both the teacher and student. For instance, one part of the model (e.g. later layers) is used to distill knowledge to another part (e.g. earlier layers). This 'teacher' part of the model may be deeper, containing more parameters, in comparison to the more shallow 'student' part of the model.

Architecture There are a variety of teacher-student architectures used for knowledge distillation. Most commonly, the teacher model will be larger and more complex in comparison to the student model. While the exact model types may vary for the two, generally the student model will have: fewer layers or fewer neurons (Wang et al., 2018a), more basic operations (Howard et al., 2017), a quantized version of the larger model (Polino et al., 2018), and so forth.

Algorithm The most basic method for transferring knowledge from the teacher to the student is to try to match the learned representations between the two models. Additional methods that move into more complex techniques, as outlined by Gou et al. (2021), include *adversarial distillation* (Wang et al., 2018b), incorporating adversarial learning inspired by generative adversarial networks (GANs) (Goodfellow et al., 2014), *multi-teacher distillation* (Hinton et al., 2015), using multiple teacher models, *cross-modal distillation* (Gupta et al., 2016), where the modality of the training data used for the teacher model is different from that of the student model (e.g. RGB images vs. depth images), as well as several other methods, such as *graph-based distillation* and *attention-based distillation*.

Developing Efficient Models Overall, knowledge distillation is one of the several techniques that are used for developing efficient models. Some of these other techniques include (1) parameter pruning, as well as parameter sharing, and (2) low-rank factorization. In the case of (1), certain parameters are removed from a large neural network, with the goal of retaining most of the original model's performance. This includes techniques such as quantization (Wu et al., 2016), binarization (Courbariaux et al., 2015), and parameter sharing (Han et al., 2015). For (2), matrix decomposition is used to remove less crucial parameters with respect to a model's performance (Denton et al., 2014).

4.4 Experimental Setup

4.4.1 Tasks and Datasets

We run experiments on five standard NLP tasks in English, with seven common datasets in total. More details regarding tasks that are considered standard within the field of NLP are in Section 2.3.2. Our first task is *sentiment analysis (SA)*. As the dataset for this task, we use movie reviews from the Internet Movie Database (IMDb) (Maas et al., 2011), in which each review is associated with a *positive* or *negative* label.

The next task is *natural language inference (NLI)*, using the Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015). The dataset consists of sentence pairs, labeled as one of three possible classes: *entailment*, *contradiction*, and *neutral*.

We then explore two separate tasks for the problem of sequence tagging. The first of these is *named entity recognition (NER)*, in which the primary goal of a model is to identify *named entities* in a given document, i.e. phrases which consist

of the names of people, locations, as well as organizations. We use two datasets for NER, the first being the very common CoNLL-2003 dataset (Tjong Kim Sang and De Meulder, 2003), consisting of data from the CoNLL-2003 shared task. Next, we use the WikiANN dataset (Pan et al., 2017), which is an NER dataset originally consisting of annotated Wikipedia articles from 282 languages, of which we use the English subset.

The second sequence tagging task is *part-of-speech (POS) tagging*. Here we again use two different datasets, both of which are dependency treebanks, i.e. corpora that are syntactically annotated based on the formalisms of Dependency Grammar (Tesnière, 1959). The first of these is the Georgetown University Multilayer (GUM) Corpus (Zeldes, 2017). This is a dataset which consists of a variety of English texts, including news, fiction, interviews, and so forth. The texts in this dataset contain several annotation categories that are standard for a dependency treebank, including part of speech for each individual token. The second POS tagging dataset is the Universal Dependencies English Web Treebank (UD EWT) corpus (Silveira et al., 2014), consisting of texts from five various web media genres, including weblogs, emails, and reviews. Both datasets utilize the Universal Dependencies set of POS tags (Petrov et al., 2012; Nivre et al., 2020), as we describe in Section 2.3.2 of Chapter 2. By including these two sequence tagging tasks, with two datasets each, we hope to examine and understand in more detail the surprisingly good results on CoNLL-2003 in Jana and Biemann (2021).

Finally, our last task is *question answering (QA)*, more specifically *extractive question answering*, in which the model is tasked with finding an answer span to a given question, from a context document. For this task we use the Stanford Question Answering (SQuAD) 2.0 dataset (Rajpurkar et al., 2018), which is a combination of the earlier SQuAD 1.1 dataset (Rajpurkar et al., 2016) and an additional set of unanswerable questions.[1] These additional unanswerable questions are in part motivated by the previous success on SQuAD 1.1, which the authors state does not represent true language understanding, since a model can simply select a span from the context document that appears to be the most similar to the question.

We present a summary of the statistics of each dataset in Table 4.1. In addition, we show more detailed statistics on the distribution of all POS tags for the GUM and EWT datasets in Table 4.2, as well as the distribution of all named entities for the CoNLL-2003 and WikiANN datasets in Table 4.3. These will be relevant for our provided error analysis to the results in Section 4.5.3.

4.4.2 Models and Training Strategies

We utilize two primary base models, with five different training and fine-tuning strategies. For our simple baseline and comparability with previous work (Jana and Biemann, 2021), we use (1) the bidirectional LSTM (BiLSTM) architecture.

[1] We use the official evaluation script, described in https://worksheets.codalab.org/worksheets/0x8212d84ca41c4150b555a075b19ccc05/

Task	Dataset	Size	Classes
SA	IMDb	50k documents	2
NLI	SNLI	570k pairs	3
NER	CoNLL-2003	~300k tokens	9
NER	WikiANN	~320k tokens	7
POS	GUM	~150k tokens	17
POS	EWT	~254k tokens	17
QA	SQuAD 2.0	150k questions	⋆

Table 4.1: Statistics for each dataset. ⋆ SQuAD contains 100k answerable and 50k unanswerable questions, with answers for the former category being a span from a corresponding text passage.

	GUM		EWT	
	Train	Test	Train	Test
NOUN	17,873	2,942	34,781	4,132
PUNCT	13,650	1,985	23,679	3,106
VERB	10,957	1,647	23,081	2,655
PRON	7,597	1,128	18,577	2,158
ADP	10,237	1,698	17,638	2,018
DET	8,334	1,347	16,285	1,896
PROPN	7,066	1,230	12,946	2,076
ADJ	6,974	1,116	12,477	1,693
AUX	4,791	719	12,343	1,495
ADV	4,180	602	10,548	1,225
CCONJ	3,247	587	6,707	739
PART	2,369	335	5,567	630
NUM	2,096	333	3,999	536
SCONJ	2,095	251	3,843	387
X	244	24	847	139
INTJ	392	87	688	120
SYM	156	35	599	92

Table 4.2: The frequency of each tag for the GUM and EWT POS tagging datasets, shown for their respective training and test splits.

We then use a pre-trained `bert-base`[2] for our remaining models, with a variety of fine-tuning methods: (2) LSTM on top of our frozen BERT encoder (`Tr/No/LSTM`), (3) softmax layer on top of the frozen BERT model (`Tr/No`), (4) last two layers of BERT fine-tuned, with the remaining parameters frozen (`Tr/Last2`), as well as (5) full BERT model fine-tuned, with the exception of the input embedding layer (`Tr/All`).

In addition to the above set of transformer configurations, we also explore these

[2] Available from https://huggingface.co/bert-base-cased.

	CoNLL-2003		WikiANN	
	Train	Test	Train	Test
O	170,524	38,554	85,665	42,879
B-PER	6,600	0	9,345	4,649
I-PER	4,528	2,773	15,085	7,721
B-ORG	6,321	5	9,910	4,974
I-ORG	3,704	2,491	23,668	11,825
B-LOC	7,140	6	10,081	5,023
I-LOC	1,157	1,919	13,664	6,661
B-MISC	3,438	9	–	–
I-MISC	1,155	909	–	–

Table 4.3: The frequency of each named entity for the CoNLL-2003 and WikiANN NER datasets, shown for their respective training and test splits.

same configurations (2-5) with the XtremeDistilTransformer (XDT)[3] model (Mukherjee et al., 2021), outlined in more detail below. This model contains far fewer parameters than the larger BERT model, with 22 million for XDT vs. 110 million for `bert-base`. In the private setting, this lower number of parameters is expected to be more favorable with respect to performance, since the ℓ_2-norm of the calculated noise for DP-SGD grows in proportion to the gradient size. This has also been noted in Abadi et al. (2016b, Sec 5.2), and various research has investigated this point in more detail (Yu et al., 2021; Tramer and Boneh, 2021; De et al., 2022), with some interestingly reporting no notable decrease in performance as the model size is scaled up (e.g. Abadi et al. (2016b); De et al. (2022)), despite the greater noise intensity. By comparing the performance of BERT and XDT, we can examine this in more detail within the NLP domain.

Additionally, the XDT model is beneficial to use for reasons of computational efficiency, with significant overhead in the private setting compared to non-private, for instance as outlined in Subramani et al. (2021). This can especially be useful when training a model on an academic budget. We provide further discussion on the efficiency of DP models in Section 4.5.6, as well as strategies for incorporating differential privacy in the academic setting in Chapter 6. We outline the XDT model in more detail below, followed by a description of our model evaluation, hyperparameter tuning and privacy settings in Sections 4.4.3, 4.4.4 and 4.4.5, respectively.

XtremeDistilTransformer (XDT) model

Knowledge distillation for the BERT model (Devlin et al., 2019) was initially proposed with the DistilBERT model (Sanh et al., 2019). Taking a `bert-base` model as the teacher model in the distillation process, it reduced the number of parameters from 110 million, down to 66 million, with authors reporting about 95% performance retention from the original BERT on the GLUE benchmark (Wang et al., 2019). Other distilled versions of BERT subsequently appeared, including TinyBERT (Jiao

[3] Available from `https://huggingface.co/microsoft/xtremedistil-l6-h384-uncased`.

et al., 2020) and MiniLM (Wang et al., 2020b), both also with 66 million parameters, as well as the XtremeDistilTransformer (XDT) model (Mukherjee et al., 2021), with only 22 million parameters. With the smaller set of parameters, reported high performance and computation speedup (5.3 times faster than `bert-base`), we opt for this last model to use in our experiments.

The overall procedure of XDT combines several knowledge distillation strategies from a larger BERT, or Electra (Clark et al., 2020) teacher model, to a more shallow student model. These include transferring knowledge from hidden representations and attention states from multiple layers, Singular Value Decomposition (SVD) for dimensionality reduction of teacher word embeddings to project them into a lower dimensional space for the student model, as well as progressive knowledge transfer (Sun et al., 2020; Mukherjee and Hassan Awadallah, 2020). The last is a technique of training different sets of parameters of the student model in stages by means of progressive freezing and unfreezing, as opposed to transferring knowledge from the entire teacher model at once.

4.4.3 Evaluation

We evaluate the performance of each model, on each task and dataset, using the macro-averaged F_1 score. With this method, results are averaged over independent F_1 calculations for each class, treating the contributions of each class equally. This is in contrast to the micro-averaged F_1 score, which first aggregates the number of true positives, false positives, and false negatives, for all classes, computing the final F_1 score from these combined values. As we discuss in Section 4.5.3, the choice of the macro-average F_1 score is very important with the presence of class imbalances in the data, reporting any negative impacts of this imbalance in the final calculated value.

4.4.4 Hyperparameter Tuning

For our hyperparameters, we primarily tune the learning rate, based on the validation set results of each dataset. We optimize for learning rates in the range $[10^{-5}, 0.1]$. For our differentially private models, we find the best learning rate at $\varepsilon = 1.0$, which we then use for other privacy budgets (e.g. $\varepsilon = 2.0$ and $\varepsilon = 5.0$). Next, we set the batch size to 32, but reduce it by a power of 2 if we encounter out of memory issues in the differentially private setting. Finally, for the BiLSTM models we use 2 layers with hidden units in the range $[128, 384]$ and an embedding dimensionality in the range $[100, 300]$, as well as a dropout of 25%.

4.4.5 Privacy Settings

As we describe in earlier chapters (e.g. Section 3.6.3 of Chapter 3), the *randomized response* mechanism of Warner (1965) can be analyzed as satisfying ($\varepsilon \approx 1.1$)-differential privacy when using fair coin flips, which can be considered as a strong privacy guarantee. We therefore carry out our experiments with the lowest privacy budget at $\varepsilon = 1.0$. In addition, to investigate the privacy/utility trade-off in more

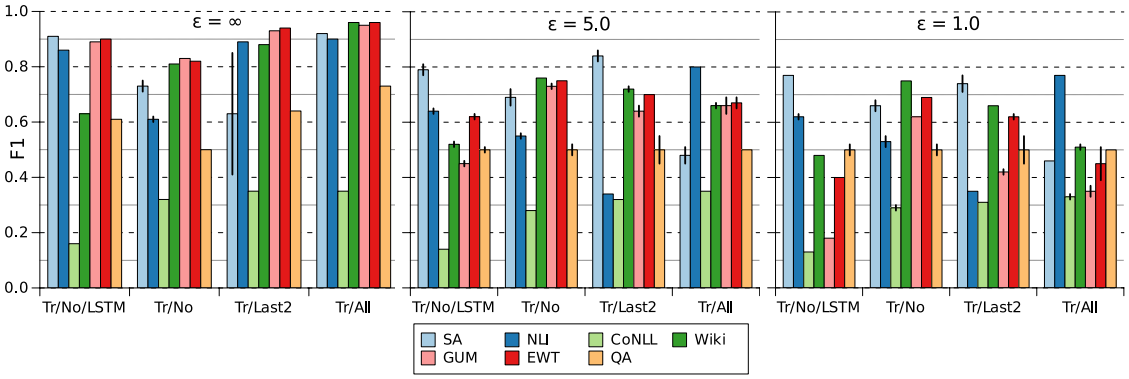


Figure 4.2: Macro-averaged F_1 scores for the BERT model in the non-private $\varepsilon = \infty$ and two of the private configurations ($\varepsilon \in \{5, 1\}$), grouped by a particular fine-tuning regime (x-axis). Each column represents the score for a specific task performed by the corresponding model. When analyzing one task (i.e. one column) in the non-private $\varepsilon = \infty$ setting for different models, the macro-averaged F_1 increases when adding fine-tuning. For the models with DP-SGD, no clear pattern can be observed, with the best model being task specific. We additionally present a complimentary diagram in Figure 4.3, depicting task-specific performance drops, discussed in Section 4.5.5.

detail for the various models and NLP tasks, we also run experiments at $\varepsilon = 2.0$ and $\varepsilon = 5.0$. We set δ to 10^{-5} and the clipping constant C to 1.0 throughout our experiments.

We implement our experiments using the PyTorch library (Paszke et al., 2019) for general-purpose deep learning, as well as the Opacus library (Yousefpour et al., 2021) for the privacy-specific parts of the implementation. As for the accounting procedure, we again employ the provided command-line tool from the TensorFlow Privacy library (Abadi et al., 2016a; Google, 2018), as described in Section 3.5.4 of Chapter 3.

4.5 Results and Analysis (BERT)

We first present our analysis for the BERT models below, with results shown in Figure 4.2. As we will see, one of the main differences between BERT and XDT is that the latter does not show a significant drop for the various private configurations, as opposed to BERT. The baseline LSTM model showed the worst performance in all setups, so we focus our analysis on the transformer-based models. For our privacy budgets, we investigate in detail three ε values of 1.0, 5.0, and ∞. The last represents the non-private setup, while the other two values show strong and weaker privacy guarantees, respectively. We report all results as macro-averaged F_1 scores, as we describe above. Additionally, we discuss points of efficiency and scalability in Section 4.5.6.

4.5.1 Sentiment Analysis

For the first task of sentiment analysis, we can see on the left of Figure 4.2 that all non-private model results show good performance. In addition, apart from the `Tr/All` setting of fully fine-tuning BERT with privacy, we see only a small performance drop for each of our model configurations.

Why does fully fine-tuning BERT fail with DP-SGD? For the non-private setting, we can see that the fully fine-tuned model, `Tr/All`, performs the best overall (e.g. 0.92 F_1 score, vs. 0.63 for `Tr/Last2`, 0.73 for `Tr/No`, and 0.91 for `Tr/No/LSTM`). In the private setting, however, there is a significant drop in F_1 score, both for $\varepsilon = 1.0$ and $\varepsilon = 5.0$. Despite the fact that the training data of the IMDb dataset is well-balanced for the positive and negative classes, the model with DP predicts everything as the negative class. With noise added to the model, one would rather expect more random predictions for each of the two balanced classes, which is not the case here. The final F_1 scores for `Tr/All` with $\varepsilon = 1.0$ and $\varepsilon = 5.0$ are 0.46 and 0.48, respectively. We can see this pattern in the confusion matrix, shown in Table 4.4, in which the vast majority of the positive class predictions are incorrect, with a lot of resulting false negatives. We can also see how well-balanced the two classes are in the dataset, with 12499 and 12500 documents for the positive and negative classes, respectively.

↓ True → Pred	Positive	Negative
Positive	**2**	12497
Negative	3	**12497**

Table 4.4: Confusion matrix for the `Tr/All` BERT configuration at $\varepsilon = 1.0$ on the task of sentiment analysis.

↓ True → Pred	Positive	Negative
Positive	**21838**	3162
Negative	3863	**21137**

Table 4.5: Confusion matrix for the `Tr/No/LSTM` BERT configuration at $\varepsilon = 1.0$ on the task of sentiment analysis.

In contrast to these results, the other fine-tuning settings (`Tr/Last2`, `Tr/No`, and `Tr/No/LSTM`) perform far better. For example, in the `Tr/Last2` configuration, fine-tuning just the last two layers of the BERT model results in an F_1 score of 0.74 for $\varepsilon = 1.0$ and 0.84 for $\varepsilon = 5.0$. We can clearly see this behavior for the best-performing private model `Tr/No/LSTM` at $\varepsilon = 1.0$, in the confusion matrix of Table 4.5. In contrast to Table 4.4, here the model shows far more correct predictions, with the majority of values on the diagonal of the table, representing the 'true positive' and 'true negative' results.

Overall, these results are consistent with previous observations (e.g. Rogers et al. (2020, Sec. 4.3) that semantic information is spread out across the full BERT model,

with the upper layers being more task-specific. By fine-tuning the full model, we are modifying the earlier more 'local' syntactic and semantic features which may be necessary for a proper prediction in sentiment analysis (Madasu and Anvesh Rao, 2019), adding a significant amount of noise to those representations.

4.5.2 Natural Language Inference

Unlike for sentiment analysis, the results for NLI show quite a different pattern. For the setting without privacy, the best models are `Tr/Last2` and `Tr/All`, both being the BERT model configurations with fine-tuning. For the setting with privacy, the `Tr/Last2` model is the worst overall, for both privacy budgets $\varepsilon = 1.0$ and $\varepsilon = 5.0$. To understand this in more detail, we need to investigate the confusion matrix for this model configuration. We present this in Table 4.6. We can see that the `Tr/Last2` model mixes up the three NLI classes of *entailment*, *contradiction* and *neutral*. It especially misses the correct predictions for the *neutral* entailment class, mostly predicting one of the other two.

One explanation for this is that the added noise in the DP setting has a negative impact on the upper, more task-specific layers of the BERT model. This then causes the model to underperform in the more complex task of NLI, not properly dealing with cross-sentence linguistic and common-sense reasoning. In addition, as described by Gururangan et al. (2018), the model may be picking up on certain artifacts in the dataset which act as linguistic cues that are far easier to recognize, instead of the task itself.

For example, the authors noticed that in the SNLI and MultiNLI datasets (Williams et al., 2018), hypotheses associated with the *entailment* class tend to have generic words in them, such as 'animal' vs. 'dog', or 'instrument' vs. 'guitar'. Similarly, hypotheses associated with the *contradiction* class tend to have negation words in them, including 'no', 'nothing', or 'never'. The model may thus have picked up on some of these easier patterns, such as the above for the entailment and contradiction classes, predicting those classes based on the artifacts, even in the case of a different true class label.

In contrast to the `Tr/Last2` model in the private setting, the `Tr/All` model shows the best performance with DP-SGD. This may indicate that training with the noisy gradient for the full BERT model's parameters increases robustness for this downstream task. Unlike in the case of sentiment analysis, here the model would not depend as much on the more local features, such as word n-grams.

4.5.3 NER and POS Tagging

The distribution of classes for the tasks of sentiment analysis and NLI is well balanced. In contrast, the distribution of classes for all four datasets of the sequence tagging tasks of NER and POS tagging is heavily imbalanced, as depicted in Tables 4.2 and 4.3, respectively (e.g. with the O tag significantly dominating the NER datasets). The impact of this imbalance can be seen in the results of all private models, as we describe below.

	↓ True → Pred	Entailment	Contradiction	Neutral
Tr/Last2	Entailment	**1356**	1699	313
	Contradiction	1122	**1780**	335
	Neutral	1217	1677	**325**
Tr/All	Entailment	**2832**	129	407
	Contradiction	272	**2530**	435
	Neutral	375	604	**2240**

Table 4.6: Confusion matrices for the `Tr/Last2` (top half) and `Tr/All` (bottom half) BERT models at $\varepsilon = 1.0$ on the task of NLI.

Overall, this effect, in which a smaller class suffers in performance, is documented within the DP community. For example, Farrand et al. (2020) look into various degrees of class imbalance and how this affects results for models trained with DP-SGD, at different privacy budgets. They find that, even with looser privacy guarantees and only slight degrees of imbalance, the impact on performance for the less represented class can be very significant, increasing with more private training steps. They hypothesize that this may be related to the gradient clipping operation within the DP-SGD algorithm, in which data points associated with smaller classes will have fewer examples in a given mini-batch, with resulting higher gradients that are more likely to be clipped. With further training, this may result in a snowball effect, in which the model improves in performance for data points associated with larger classes, with an increasing number of data points for the smaller classes becoming outliers. Similarly, Bagdasaryan et al. (2019) investigated class imbalance with DP-SGD and showed that, if there is a degree of unfairness exhibited in a model with respect to less represented classes, this is amplified in the private setting.

We see this behavior in our models trained with DP for both the tasks of NER and POS tagging. Only the most common tags are predicted with a decent level of performance, while the other tags suffer misclassifications. We can see this more clearly in Tables 4.7 and 4.8, for NER and POS tagging, respectively, showing F_1 scores for each class using the BERT `Tr/All` model at $\varepsilon = 1.0$. For example, the O tag for NER achieves 0.98 and 0.86 F_1 scores for the CoNLL-2003 and WikiANN datasets, respectively. In contrast, nearly all other NER tags have around 0.00 F_1 score for CoNLL-2003, apart from the I-PER class. Similarly, the tags for nouns, punctuation, verbs, pronouns, adpositions, and determiners in POS tagging show far better performance than all the rest. For both the GUM and EWT datasets, these tags are above F_1 scores of 0.60, for instance 0.66 for the NOUN tag of the GUM dataset, or 0.87 for the PUNCT tag of the EWT dataset. Apart from the AUX tag, all of the other tags not highlighted in Table 4.8 have F_1 scores below 0.20, with six of them being at an F_1 score of 0.00 for both datasets.

Drawing conclusions using unsuitable metrics?

The average over F_1 scores for all classes, i.e. the macro-averaged F_1, suffers due to the incorrect predictions for the imbalanced classes. Despite this, micro-averaged

	CoNLL-2003	WikiANN
O	0.98	0.86
B-PER	0.00	0.69
I-PER	0.67	0.57
B-ORG	0.00	0.14
I-ORG	0.01	0.54
I-LOC	0.00	0.46
B-LOC	0.00	0.44
B-MISC	0.00	–
I-MISC	0.00	–

Table 4.7: F_1 scores for each individual class on the NER task for the CoNLL-2003 and WikiANN datasets, using the BERT `Tr/All` model at $\varepsilon = 1.0$. All classes are predicted with very poor performance, with the exception of the O tag for both datasets (highlighted).

	GUM	EWT
NOUN	0.66	0.62
PUNCT	0.85	0.87
VERB	0.64	0.72
PRON	0.65	0.72
ADP	0.73	0.80
DET	0.81	0.83
PROPN	0.17	0.16
ADJ	0.13	0.03
AUX	0.41	0.69
ADV	0.00	0.10
CCONJ	0.06	0.02
PART	0.00	0.00
NUM	0.00	0.00
SCONJ	0.00	0.00
X	0.00	0.00
INTJ	0.00	0.00
SYM	0.00	0.00

Table 4.8: F_1 scores for each individual class on the POS tagging task for the GUM and EWT datasets, using the BERT `Tr/All` model at $\varepsilon = 1.0$. Acceptable performance only appears for the more common tags, in the highlighted rows.

F_1 scores are not negatively impacted. For example, while the macro-F_1 score in the above example for CoNLL-2003 is only 0.18 for the BERT `Tr/All` model at $\varepsilon = 1.0$, the micro-F_1 score is 0.85! This is due to the fact that the macro-averaged F_1 score treats all classes as being equally important in the final score calculation, while micro-averaged F_1 is biased with regards to the frequency of classes. We therefore suggest using the macro-averaged F_1 score to evaluate models trained

with differential privacy, especially when dealing with imbalanced datasets. The significant difference in the macro- and micro-F_1 scores explains the unintuitive invariance of NER to the DP setting in Jana and Biemann (2021).

Non-private NER models misclassify the tag prefix, but NER models with DP fail on the tag type itself

When we look into the NER results in more detail, we notice an interesting pattern with respect to the tags that the model misclassifies. In the non-DP setting, the model generally classifies the type of tag correctly (e.g. LOC, PER, ORG, MISC), but occasionally misclassifies the position itself, i.e. the I and B prefixes. We can see this in the confusion matrix of Table 4.9, in which tokens associated with the I-LOC tag are very often misclassified as the B-LOC tag, with 1558 such misclassifications vs. only 232 true positives for I-LOC. The situation is very similar for I-PER vs. B-PER, I-ORG vs. B-ORG, as well as I-MISC vs. B-MISC, all highlighted in the table.

In the setting with differential privacy, we can see that the misclassifications of the model are further impacted. In addition to the position of a tag, the type of tag itself is also misclassified. This can be seen in more detail in Table 4.10. For example, I-MISC is incorrectly predicted as B-LOC 502 times, while I-ORG is incorrectly predicted as B-LOC 763 times.

↓ True → Pred	O	B-PER	I-PER	B-ORG	I-ORG	B-LOC	I-LOC	B-MISC	I-MISC
O	**38207**	26	3	47	41	19	9	66	80
B-PER	0	**0**	0	0	0	0	0	0	0
I-PER	17	1556	**1150**	23	10	15	1	1	0
B-ORG	0	0	0	**0**	5	0	0	0	0
I-ORG	42	27	3	1514	**767**	53	29	42	14
B-LOC	0	0	1	0	2	**0**	1	0	2
I-LOC	22	7	3	49	12	1558	**232**	30	5
B-MISC	0	0	0	0	0	0	0	**4**	5
I-MISC	51	9	3	32	9	23	4	594	**183**

Table 4.9: Confusion matrix for the `Tr/All` BERT model without DP on the CoNLL-2003 dataset. We can see that sometimes the model misclassifies the position of a tag (e.g. B or I prefix) (highlighted), but the tag itself (e.g. PER, ORG, etc.) is correctly classified most of the time.

NER models are majority-voting with meaningful ε values already, and become random only at very low ε values

As we described in Section 4.2, Jana and Biemann (2021) reported that a bidirectional LSTM trained with differential privacy on NER has almost no difference in accuracy, in comparison to the non-private model. Our experiments, however, reveal that even with $\varepsilon = 1.0$, the models trained with DP predict the outside O tag nearly every time, shown in the confusion matrix of Table 4.11 for the bidirectional LSTM model. As we mention above, the micro-F_1 score does not actually change between the DP and non-DP settings, due to the imbalanced nature of the dataset, in which the O tag is the majority label. However, the macro-F_1 score shows a more

↓ True → Pred	O	B-PER	I-PER	B-ORG	I-ORG	B-LOC	I-LOC	B-MISC	I-MISC
O	**37540**	61	38	149	0	304	0	0	0
B-PER	0	**0**	0	0	0	0	0	0	0
I-PER	33	1057	**1521**	77	0	48	0	0	0
B-ORG	0	0	0	**4**	0	1	0	0	0
I-ORG	192	90	142	1157	**9**	763	0	0	0
B-LOC	0	1	2	2	0	**1**	0	0	0
I-LOC	58	14	38	169	0	1577	**0**	0	0
B-MISC	1	2	0	1	0	5	0	**0**	0
I-MISC	178	21	14	110	0	502	0	0	**0**

Table 4.10: Confusion matrix for the `Tr/All` BERT model at $\varepsilon = 1.0$ on the CoNLL-2003 dataset. Here we can see that the model incorrectly predicts both the tag position and the type of tag itself (examples highlighted).

accurate evaluation of these models, revealing the actual misclassifications, as seen in the case described for the BERT `Tr/All` model, in Table 4.7 above.

While such a discrepancy between the two scores exists for models in the non-private setting, it is evident that, with more model complexity (e.g. BERT), the problem is less severe. As we tried smaller and smaller values of ε, we noticed that this behavior continues to stay the same, up until we reach extremely low privacy budgets. We only obtain a different model behavior at a value of $\varepsilon = 0.00837$. In this case, the model ends up seemingly predicting everything randomly, as seen in Table 4.12, with model predictions for each target label very spread out.

↓ True → Pred	O	B-PER	I-PER	B-ORG	I-ORG	B-LOC	I-LOC	B-MISC	I-MISC
O	**41021**	9	6	5	9	4	52	4	32
B-PER	0	**0**	0	0	0	0	0	0	0
I-PER	2821	1	**0**	0	0	0	2	0	3
B-ORG	5	0	0	**0**	0	0	0	0	0
I-ORG	2524	1	1	0	**0**	0	3	1	2
B-LOC	6	0	0	0	0	**0**	0	0	0
I-LOC	1935	1	0	1	0	0	**0**	0	1
B-MISC	9	0	0	0	0	0	0	**0**	0
I-MISC	1013	0	0	0	0	0	1	0	**1**

Table 4.11: Confusion matrix for the bidirectional LSTM model at $\varepsilon = 1.0$ on the CoNLL-2003 dataset. The model overwhelmingly predicts the outside O tag (highlighted).

4.5.4 Question Answering

Our last task is possibly one of the most challenging for the models trained with DP-SGD. In the non-private setting, models with an increasing number of fine-tuned layers show improvements in results (e.g. F_1 score of 0.50 for `Tr/No` vs. 0.64 for `Tr/Last2` vs. 0.73 for `Tr/All`). With differential privacy, however, performance for all models is reduced to an F_1 score of 0.5, disregarding the strictness of the privacy guarantee ($\varepsilon = 1.0, 2.0, 5.0$). Nearly all questions in the SQuAD 2.0 dataset were

↓ True → Pred	O	B-PER	I-PER	B-ORG	I-ORG	B-LOC	I-LOC	B-MISC	I-MISC
O	**3464**	4161	21208	144	170	1993	3190	722	6035
B-PER	0	**0**	0	0	0	0	0	0	0
I-PER	234	348	**1440**	3	20	140	226	34	377
B-ORG	2	0	2	**0**	0	1	0	0	0
I-ORG	271	310	1161	8	**9**	115	184	39	430
B-LOC	0	1	4	0	0	**0**	1	0	0
I-LOC	130	252	1032	2	11	84	**153**	29	243
B-MISC	1	1	4	0	0	3	0	**0**	0
I-MISC	76	100	561	3	4	49	73	17	**132**

Table 4.12: Confusion matrix for the bidirectional LSTM model at $\varepsilon = 0.00837$ on the CoNLL-2003 dataset. The model predicts tags seemingly at random at this level of privacy (highlighted).

predicted as *unanswerable* for every model with DP-SGD. Due to the fact that 50% of the test set has this *unanswerable* label, the model is able to achieve this 0.5 F_1 score by always predicting it.

Overall, we can see similar behavior for question answering, as for the NER and POS tagging tasks. Question answering has many possible output classes in the span prediction process, being a relatively challenging task. We suggest to run experiments on the SQuAD 1.0/1.1 versions of the dataset, in order to look into this further in future analyses of the impacts of model training with DP-SGD on NLP tasks. Without this category of unanswerable questions, we would be able to investigate more closely how the DP models behave on the span prediction task itself.

4.5.5 Performance drop with stricter privacy

When utilizing the DP-SGD algorithm, we expect an increase in performance drop as we decrease the privacy budget ε. While this is also what we observe throughout our experiments, there is no single consistent pattern among the various tasks and models. We can see this in more detail in Figure 4.3, which shows the impact of reducing ε on F_1 scores, complimentary to Figure 4.2, with the inclusion of results for $\varepsilon = 2.0$. As an example, the BERT `Tr/All` model shows a very large drop for sentiment analysis when moving from the non-private to the private setting for any ε value. While the F_1 score is 0.92 at $\varepsilon = \infty$, this drops down to 0.48 at $\varepsilon = 5.0$ and 0.46 at $\varepsilon = 1.0$. In contrast, the drop for NLI with this model is far smaller, with an F_1 score of 0.90 at $\varepsilon = \infty$, 0.80 at $\varepsilon = 5.0$, and 0.77 at $\varepsilon = 1.0$.

Similar discrepancies can be seen for other models and tasks. For example, in contrast to the above NLI results performing best with BERT `Tr/All`, for sequence tagging on the WikiANN dataset, the best model is `Tr/No`, with an F_1 score of 0.75 at $\varepsilon = 1.0$, remaining relatively constant with the non-private result of 0.81.

Overall, we can see that a specific model and training regime needs to be carefully selected for an intended specific privacy requirement of the training data. There is

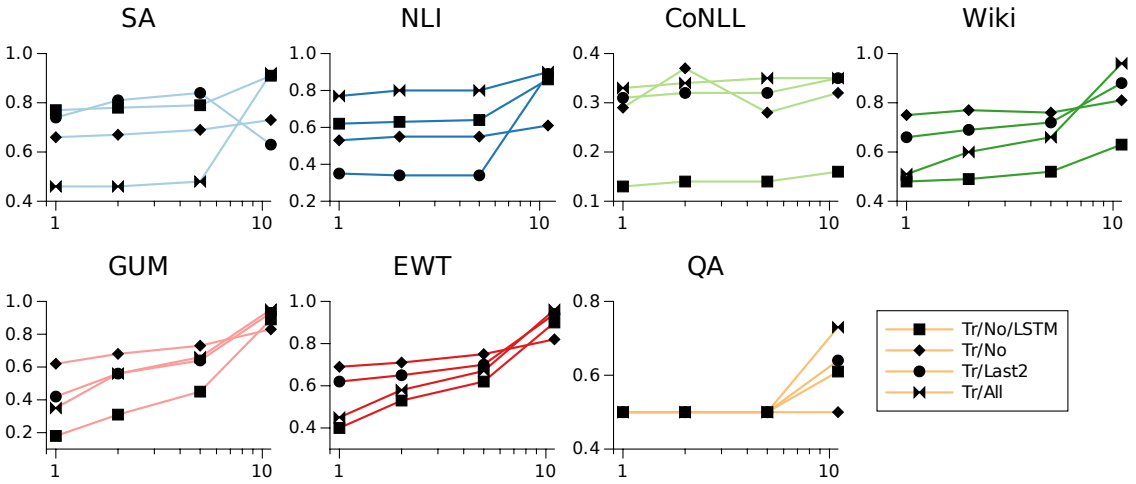


Figure 4.3: Results for all datasets and BERT configurations, with y-axis representing the macro-averaged F_1 score and x-axis representing the privacy budget $\varepsilon \in \{1, 2, 5, \infty\}$ (log scale).

no single winning strategy with regards to model selection and training regime that stands out to be used in the private setting for the various NLP tasks and datasets.

4.5.6 Efficiency and scalability of differentially private models

In addition to performance, questions of computational efficiency, such as training times, are a significant concern, especially in the current sphere of large transformer models. While the clipping and noise operations at first glance may seem relatively harmless with respect to computational efficiency, DP-SGD has been noted to significantly increase training times (e.g. Carlini et al. (2019); Thomas et al. (2020)), compared to the non-private setting. This can be explained by the fact that modern deep learning frameworks such as TensorFlow (Abadi et al., 2016a) and PyTorch (Paszke et al., 2019) do not allow for access to per-example gradients of individual data points, instead only providing the gradient averaged over a given minibatch. In order to then access a per-example gradient for the clipping step of DP-SGD, we need to process each one of them individually, without the benefits of parallel processing that these frameworks provide. This subsequently results in the significant increase of computation time for the DP-SGD setting. We refer to Subramani et al. (2021) for further discussion on this point.

To investigate these issues of computational efficiency for our experiments, we report the average time it takes for our models to complete an epoch in Table 4.13. For all experiments, we can see that the number of parameters for fine-tuning the model plays a crucial role in the degree of increase for the average epoch time, when moving from the non-DP to the DP setting. With more fine-tuned parameters, this relative increase is more significant. This pattern can be seen for all tasks, with a larger relative increase in epoch times for the `Tr/All` setting, compared to the `Tr/Last2` setting. For example, the SNLI dataset shows a relative increase of

16.40x for `Tr/Last2` and 26.54x for `Tr/All`. Perhaps most severe is the case of POS tagging with the EWT dataset, with a relative 7.02x increase for `Tr/Last2` and 41.18x increase fo `Tr/All`. In the case of a limited computational budget, the model setting with less fine-tuned parameters would be far more feasible for running differentially private model training. In addition, frameworks such as JAX (Frostig et al., 2018; Bradbury et al., 2018) may offer some solutions to these issues of longer computation times for the DP-SGD setting, as demonstrated in Subramani et al. (2021); Yin and Habernal (2022), with various features such as Just-In-Time (JIT) compilation of the Accelerated Linear Algebra (XLA) compiler (Google) leading to lower runtimes.

Task	Config	Epoch time (no DP)	Epoch time (DP)	Increase with DP
SA	`Tr/Last2`	4 m 22 s	0 h 10 m 32 s	2.41x
SA	`Tr/All`	9 m 07 s	0 h 47 m 08 s	5.17x
NER (CoNLL-2003)	`Tr/Last2`	0 m 26 s	0 h 02 m 24 s	5.54x
NER (CoNLL-2003)	`Tr/All`	0 m 57 s	0 h 27 m 55 s	29.39x
NLI	`Tr/Last2`	13 m 15 s	3 h 37 m 15 s	16.40x
NLI	`Tr/All`	22 m 32 s	9 h 57 m 57 s	26.54x
POS Tagging (EWT)	`Tr/Last2`	0 m 53 s	0 h 06 m 12 s	7.02x
POS Tagging (EWT)	`Tr/All`	1 m 12 s	0 h 49 m 25 s	41.18x
QA	`Tr/Last2`	12 m 21 s	1 h 57 m 43 s	9.53x
QA	`Tr/All`	44 m 07 s	11 h 05 m 15 s	15.08x

Table 4.13: Average epoch time of training a model for each NLP task, using the BERT `Tr/Last2` and `Tr/All` configurations for the partially fine-tuned and fully fine-tuned BERT models, respectively. Rightmost column shows the degree of increase from the non-private to the private setting.

4.6 XtremeDistilTransformer Model

In addition to our experiments using the `bert-base` model above, we run the same configurations (i.e. fine-tuning regimes) for all of our datasets using the XtremeDistilTransformer model, outlined in Section 4.4.2. We present our analysis of these results below, shown in Figures 4.4 and 4.5.[4]

4.6.1 Comparing non-private XDT and BERT

In the non-differentially private setting, both the **XDT and BERT models behave in a similar manner**. For example, results for the `Tr/All` configuration are mostly the same for both models, for the majority of datasets, with the exception of the two NER datasets. The task of sentiment analysis with `Tr/All` shows results of 0.92 F_1 score for both BERT and XDT. For NLI, we see 0.90 F_1 with BERT and 0.89 F_1 with XDT. In POS tagging, both BERT and XDT reach an F_1 score of

[4] Due to some limitations in further GPU computation capacities at the time of running experiments, we did not complete the QA task for this model, which we omit in the below results.

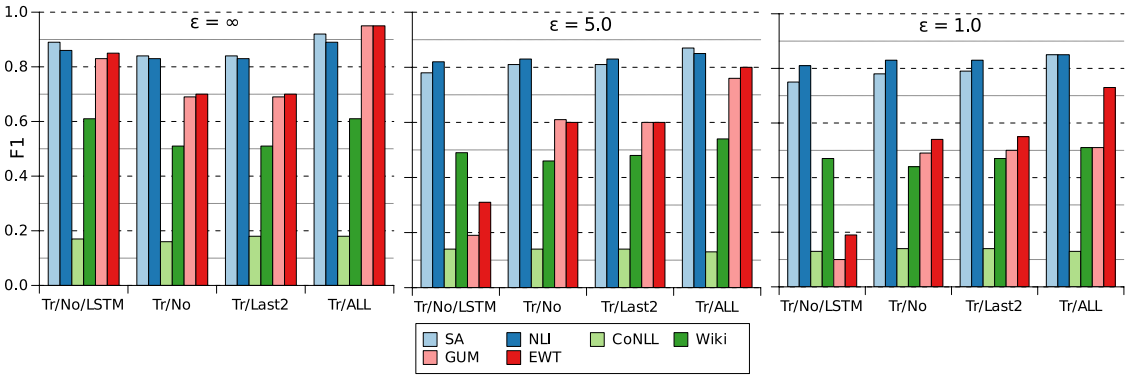


Figure 4.4: Macro-averaged F_1 scores for the XDT model in the non-private $\varepsilon = \infty$ and two of the private configurations ($\varepsilon \in \{5, 1\}$), grouped by a particular fine-tuning regime (x-axis). Each column represents the score for a specific task performed by the corresponding model. When analyzing one task (i.e. one column) in the non-private $\varepsilon = \infty$ setting for different models, the macro-averaged F_1 tends to increase when adding fine-tuning. For the models with DP-SGD, generally there is either no notable drop in performance (e.g. SA and NLI), or a clear drop (e.g. GUM). We additionally present a complimentary diagram in Figure 4.5, depicting task-specific performance drops.

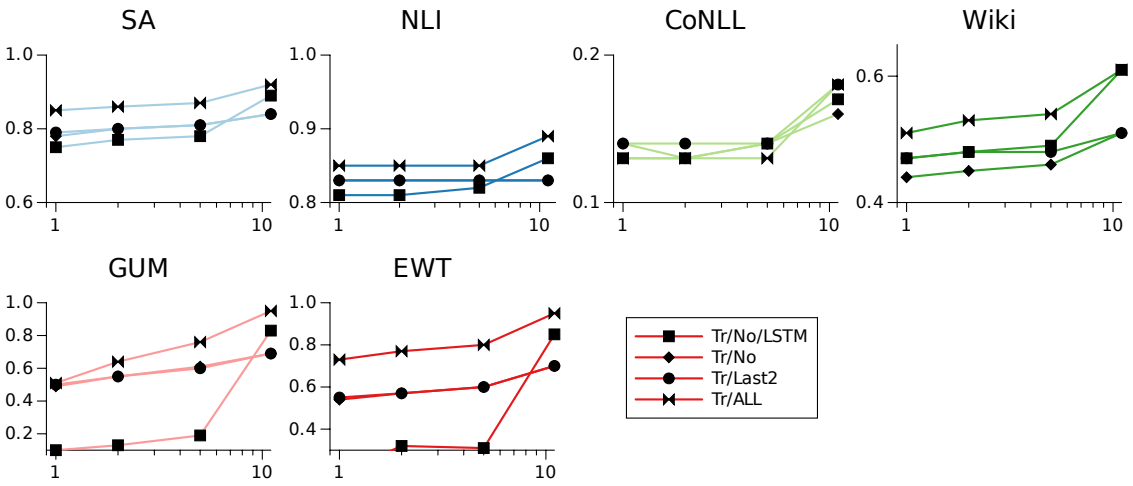


Figure 4.5: Results for all datasets and XDT configurations, with y-axis representing the macro-averaged F_1 score and x-axis representing the privacy budget $\varepsilon \in \{1, 2, 5, \infty\}$ (log scale).

0.95 on the GUM dataset, as well as 0.96 for BERT and 0.95 for XDT on the EWT dataset.

There are some differences, however, in certain settings. For sentiment analysis with the `Tr/Last2` configuration, the XDT model is much better (0.84 F_1 for XDT vs. 0.63 for BERT). For NLI with the `Tr/No` configuration, we see an improvement for XDT over BERT (0.83 vs. 0.61 F_1 score, respectively). Results for the CoNLL-2003 dataset are significantly lower for XDT than BERT. For example, the `Tr/Last2` configuration with XDT reaches 0.18 F_1 score, in comparison to 0.35 for the corresponding BERT configuration.

For the POS tagging datasets with XDT, while the `Tr/No/LSTM` and `Tr/All` configurations perform well in the non-private setting, the `Tr/No` and `Tr/Last2` configurations perform far worse. For example, XDT with the `Tr/No/LSTM` configuration reaches 0.83 F_1 score on the GUM dataset, while XDT with `Tr/No` and `Tr/Last2` both reach an F_1 score of 0.69. This is in contrast to the BERT model results, where all four configurations show more similar performance to one another (e.g. 0.89 F_1 for `Tr/No/LSTM` vs. 0.83 F_1 for `Tr/No` and 0.93 for `Tr/Last2`, for the GUM dataset). Hence, while the two non-private models are generally comparable in behavior with respect to performance, they diverge in some specific cases, primarily for the sequence tagging datasets.

4.6.2 Differentially private XDT

Looking at the XDT model in isolation, we can see two main patterns that emerge in the DP setting. **For sentiment analysis and NLI, there is almost no drop** in F_1 score, across different privacy budgets. For example, sentiment analysis with the `Tr/All` configuration only drops from 0.92 F_1 at $\varepsilon = \infty$ to 0.87 at $\varepsilon = 5.0$, and 0.85 at $\varepsilon = 1.0$. On the task of NLI, the `Tr/All` configuration only drops from 0.89 F_1 score at $\varepsilon = \infty$ to 0.85 at all other ε values. Similarly, NLI with the `Tr/Last2` configuration stays at 0.83 for all privacy budgets.

In contrast, **for the various sequence tagging tasks, there is a larger drop** from the non-private setting. This can be seen in the two POS tagging datasets, with the GUM dataset, for instance, resulting in the `Tr/All` model dropping from 0.95 F_1 with $\varepsilon = \infty$ to 0.76 F_1 with $\varepsilon = 5.0$, and further down to 0.51 F_1 with $\varepsilon = 1.0$. In addition, if results were already quite low in the non-private setting, then they expectedly remain low in the private setting (e.g. CoNLL-2003 with `Tr/All` dropping from 0.18 F_1 at $\varepsilon = \infty$ to 0.13 F_1 at both $\varepsilon = 5.0$ and $\varepsilon = 1.0$).

4.6.3 Differentially private XDT vs. BERT

Finally, we compare both the XDT and BERT models in the private setting. Here we notice one primary difference. While the **BERT model shows a general decline when fine-tuning all layers** (`Tr/All`), **XDT does not have a significant drop for all fine-tuning configurations, for most datasets** (e.g. staying nearly the same for sentiment analysis and NLI in the private setting, as mentioned above). In fact, in some cases, such as for NLI, XDT shows a better performance than

BERT. For instance, it achieves an F_1 score of 0.85 with `Tr/All` at $\varepsilon = 1.0$, vs. the BERT results of 0.77 with `Tr/All`. One explanation for this could be that, since the XDT model has fewer parameters, it therefore would have a smaller ℓ_2-sensitivity of the gradient of the loss function, as the gradient has fewer dimensions, with a corresponding lower ℓ_2-norm of the added Gaussian noise. This then leads to less noise added at each training iteration, as we discuss in more detail in Section 4.4.2. Since XDT is trained to mimic the behavior of BERT with minimal performance loss, the result is a model that shows similar performance to BERT in the non-private setting, but is less impacted by noise in the private setting.

This does not mean, however, that XDT is always the better choice. As we describe above, the sequence tagging tasks are far worse in performance for XDT. For example, the CoNLL-2003 dataset is at an F_1 score of 0.13 or 0.14 for all configurations at $\varepsilon = 1.0$ with XDT, while the BERT results mostly hover around an F_1 score of 0.30. In the case of POS tagging, BERT also often outperforms XDT. For instance, when running on the GUM dataset for the `Tr/No` configuration, XDT achieves an F_1 score of 0.49 at $\varepsilon = 1.0$, while BERT reaches 0.62. Nevertheless, with the large drop for the BERT model in the `Tr/All` configuration, XDT performs better here at $\varepsilon = 1.0$, with 0.35 F_1 vs. 0.51 F_1, respectively.

4.6.4 Summary and conclusions on XDT

Our overall conclusions on the performance of XDT are as follows. In some cases, such as for the sentiment analysis or NLI tasks, a model with a fewer set of parameters such as XDT can be beneficial over the larger BERT model in the private setting. In other cases, such as for sequence tagging tasks, however, it may be better to use the BERT model, especially for tasks such as NER.

Finally, an additional point in favor of using XDT over BERT is for running the model at faster epoch times. We present this in Tables 4.14 and 4.15 for a direct comparison between the two models in the non-DP and DP settings, respectively. The improvements in runtimes for the XDT model are notable for all fine-tuning configurations and datasets. These improvements are especially notable for the DP setting, with training times for XDT being only a small percentage of the BERT counterparts (e.g. XDT takes only 2% of the runtime that BERT takes for the `Tr/Last2` configuration on NLI). This decrease in epoch times is additionally achievable due to larger possible batch sizes with XDT, since the model requires less memory than BERT.

4.7 Chapter Summary

We have investigated the application of the DP-SGD methodology to a variety of NLP text classification tasks and models. With relation to our original research question on whether there is a systematic strategy that can be applied in the DP setting, we can clearly see that this is not the case. The best performance in the DP setting for each NLP task and dataset varies across different model configurations.

Task	Config	Epoch time (BERT)	Epoch time (XDT)	XDT vs. BERT
SA	`Tr/Last2`	4 m 22 s	0 m 21 s	0.08x
SA	`Tr/All`	9 m 07 s	0 m 52 s	0.10x
NER (CoNLL-2003)	`Tr/Last2`	0 m 26 s	0 m 16 s	0.62x
NER (CoNLL-2003)	`Tr/All`	0 m 57 s	0 m 25 s	0.44x
NLI	`Tr/Last2`	13 m 15 s	2 m 34 s	0.19x
NLI	`Tr/All`	22 m 32 s	18 m 32 s	0.82x
POS Tagging (EWT)	`Tr/Last2`	0 m 53 s	0 m 16 s	0.30x
POS Tagging (EWT)	`Tr/All`	1 m 12 s	0 m 19 s	0.26x
QA	`Tr/Last2`	12 m 21 s	6 m 45 s	0.55x
QA	`Tr/All`	44 m 07 s	7 m 24 s	0.17x

Table 4.14: Average epoch time of training a model for each NLP task without DP-SGD, using the BERT and XDT models with the `Tr/Last2` and `Tr/All` configurations, for the partially fine-tuned and fully fine-tuned BERT/XDT models, respectively. Rightmost column shows the relative speed of XDT compared to BERT. Between the two models, we can see that XDT takes far less time to train.

Task	Config	Epoch time (BERT)	Epoch time (XDT)	XDT vs. BERT
SA	`Tr/Last2`	0 h 10 m 32 s	0 m 21 s	0.03x
SA	`Tr/All`	0 h 47 m 08 s	14 m 29 s	0.31x
NER (CoNLL-2003)	`Tr/Last2`	0 h 02 m 24 s	0 m 17 s	0.12x
NER (CoNLL-2003)	`Tr/All`	0 h 27 m 55 s	1 m 34 s	0.06x
NLI	`Tr/Last2`	3 h 37 m 15 s	3 m 36 s	0.02x
NLI	`Tr/All`	9 h 57 m 57 s	62 m 32 s	0.10x
POS Tagging (EWT)	`Tr/Last2`	0 h 06 m 12 s	0 m 15 s	0.04x
POS Tagging (EWT)	`Tr/All`	0 h 49 m 25 s	1 m 32 s	0.03x
QA	`Tr/Last2`	1 h 57 m 43 s	6 m 32 s	0.06x
QA	`Tr/All`	11 h 05 m 15 s	53 m 35 s	0.08x

Table 4.15: Average epoch time of training a model for each NLP task with DP-SGD, using the BERT and XDT models with the `Tr/Last2` and `Tr/All` configurations, for the partially fine-tuned and fully fine-tuned BERT/XDT models, respectively. Rightmost column shows the relative speed of XDT compared to BERT. The difference in runtimes for the two models is especially noticeable here in the DP setting (e.g. EWT dataset).

Our experiments overall show four primary research conclusions. First, the performance of **DP-SGD is negatively impacted with skewed class distributions** that are present in many NLP tasks, with models trained using DP-SGD heavily reliant on majority classes in the dataset. Second, **different fine-tuning regimes** of transformer-based models, which in turn translates to adding noisy updates to the parameters of different layers of a transformer model, **behave differently depending on the specific task**. Among these regimes, a single approach does not generalize for best performance across various NLP tasks and datasets. There is therefore no single method that can be consistently applied with respect to

fine-tuning a model in the DP-SGD setting. The best private fine-tuning setting has to be investigated in a task-specific manner. This is in contrast to the non-private setting, in which 'bigger is generally better'. Third, **private NER has been misinterpreted in previous work due to unsuitable evaluation metrics** that do not take dataset class imbalance into account.

Finally, the fourth conclusion is that **the distilled variants of larger language models**, such as XDT vs. BERT, **can be beneficial to use for some NLP tasks in terms of performance**, although this is again task-specific and not a general rule that can be applied for all tasks and training configurations. With respect to computational efficiency, however, these distilled models are incredibly beneficial for the DP setting, with far smaller runtimes.

Chapter 5

Building a Framework for Reproducible and Transparent Differentially Private Text Rewriting

We now turn to the last part of this thesis. Having investigated the privatization of NLP models, specifically in the graph setting (Chapter 3) and more generally across a variety of models and tasks (Chapter 4), we now shift our focus to the textual data itself. In the previous two chapters we dealt with privatizing textual data with respect to a given NLP model and task, i.e. the setting of global differential privacy, as outlined in Section 2.2.1 of Chapter 2. In the current and following chapter, we now address RQ3 from Chapter 1:

RQ3 How can we successfully privatize textual data, independent from a specific NLP system?

This is the setting in which the *data itself is perturbed*, in order to obtain a given differential privacy guarantee, i.e. local differential privacy (LDP). This means that the privatized data can then be used for a variety of downstream tasks, using any corresponding downstream model architecture. As we discuss in Chapter 1, we can reword our research question as: *To what extent is local differential privacy (LDP) possible for textual data?* A setup that has been recently explored for achieving this is differentially private text rewriting, in which a document is rewritten with LDP guarantees. We will see that there are significant difficulties in reaching a decent privacy/utility trade-off due to the *strict adjacency constraint*, in which any two given documents are considered neighboring and must therefore be indistinguishable from one another, up to a selected ε privacy budget.

Before designing a model to carry out this task and pushing the boundaries for the privacy/utility trade-off that can be reached, we first address an issue that is currently present in the field of text privatization, namely the matter of transparency and reproducibility for this private text rewriting task. We tackle this problem in the current chapter, developing the `DP-Rewrite` framework, a platform for differentially

private text rewriting that is open, modular, and transparent.

5.1 Introduction

The task of text rewriting with differential privacy provides theoretical guarantees for protecting the privacy of individuals in textual documents. This setup takes a given document as input and uses a text generation model to produce an output document. The goal of this model is to output a document as close as possible to the original input. By incorporating a differential privacy component in this model, the output has probabilistic guarantees on the degree to which the author of the original input document can be de-identified, if that privatized output document is published. The strength of this guarantee is dictated by the privacy budget ε. For example, given a text "I want to fly from Newark to Cleveland on Friday", the text rewriting system may output "Flights from Los Angeles to Houston this week". In the case of training an intent classification model, such as a system for airline travel inquiries, either of these two documents would provide a useful contribution to improving the learned representations of the model.

In practice, existing text rewriting systems may lack the means to validate their privacy-preserving claims, leading to problems of transparency and reproducibility. For example, Krishna et al. (2021) proposed the ADePT text rewriting system, based on the Laplace and Gaussian mechanisms (see Section 2.2.1 of Chapter 2). However, as discussed in Habernal (2021), it turned out that their DP method was formally flawed. We also see another recent approach, DP-VAE (Weggenmann et al., 2022), which shows results that look surprisingly good for the level of guaranteed privacy. However, neither ADePT nor DP-VAE published their source codes, so the community has no means to perform any empirical checks to validate their privacy-preserving claims. Therefore, the primary obstacles to the accountability of DP text rewriting systems are the issues of transparency and reproducibility.

We thus introduce `DP-Rewrite`, an open-source framework for differentially private text rewriting which aims to solve these problems by being modular, extensible, and highly customizable. We hypothesize that by integrating different models, downstream datasets, pre-training procedures, and evaluation metrics into one software package, we can improve these issues of transparency, accountability, and reproducibility of research in differentially private text rewriting, allowing the community to obtain further insight into the utility and potential pitfalls of such systems.

Our contributions in this chapter are as follows. First, we present `DP-Rewrite` for differentially private text rewriting, outlined above. As part of the framework, we include a corrected reimplementation of the ADePT system of Krishna et al. (2021) as a baseline. We integrate pre-training on several datasets, as well as the ability to smoothly carry out downstream experiments using different privacy guarantees, adjusting the privacy budget ε. We describe this framework in detail in Section 5.3.

Second, we present a case study on the ADePT system, in which we are able to easily detect another privacy leak in its proposed methodology. Specifically, this has

to do with the pre-training strategy for the autoencoder model, in which the system memorizes the input data. This is presented in detail in Section 5.4.

5.2 Related Work and Background

Although the problem of simple data redaction is a widely researched field with several promising approaches (Hill et al., 2016; Lison et al., 2021), the related problem of private text transformation is still largely unexplored. We are working in the framework of local differential privacy (LDP), in which an input document is put through a randomized mechanism (e.g. neural network model with an incorporated DP mechanism) that perturbs it using carefully calculated DP noise, providing a privatized output document. Since we are working in the LDP setting, this results in very strict requirements of document adjacency, in which any two documents are considered *neighboring* in the DP sense, resulting in a large amount of noise that needs to be added to the input document, in order to achieve a given DP guarantee. We refer to Sections 2.2.1 and 2.2.3 for a more detailed background on the differences between global and local differential privacy, as well as a simple example of one of the most famous local differential privacy mechanisms, namely randomized response, respectively.

The following is a description of previous work on DP text rewriting. For further details on related work in differential privacy and NLP, we refer to Section 2.2.5 of Chapter 2. As we mention above, Krishna et al. (2021) proposed ADePT for this task. The system consists of an autoencoder that learns a compressed latent representation of text, and a DP rewriting component that uses the trained autoencoder, adds Laplace or Gaussian noise to the latent representation vector, and generates the privatized text. Due to a formal error in the scale of the Laplace noise, ADePT violated differential privacy (Habernal, 2021). We outline the model architecture and differential privacy module of ADePT in Section 5.4.1, and additionally discuss the system in more detail in the next chapter, Section 6.3.1.

A more recent text rewriting system is DP-VAE (Weggenmann et al., 2022), which added constraints to the vanilla VAE model latent space (Kingma and Welling, 2014) to obtain a bounded sensitivity on its mean and variance parameters. Despite the high difficulties of the task, the paper reports surprisingly high performance for high privacy standards. Since their experimental description lacks some key details and the code base is not public, we cannot reproduce their approach.

Mattern et al. (2022a) explored text rewriting with global differential privacy, sampling from a generative language model trained with DP. In addition, there are a number of word-level DP systems (Feyisetan et al., 2019; Xu et al., 2020b; Bo et al., 2021), where individual word embeddings are perturbed with DP, with new words then sampled close to these privatized vectors in the latent space. As Mattern et al. (2022b) point out, there are several shortcomings of such approaches, including a lack of obfuscating syntactic information and the inability to provide proper anonymization. In essence, these methods do not privatize a full utterance, but only single words.

5.3 Description of Software

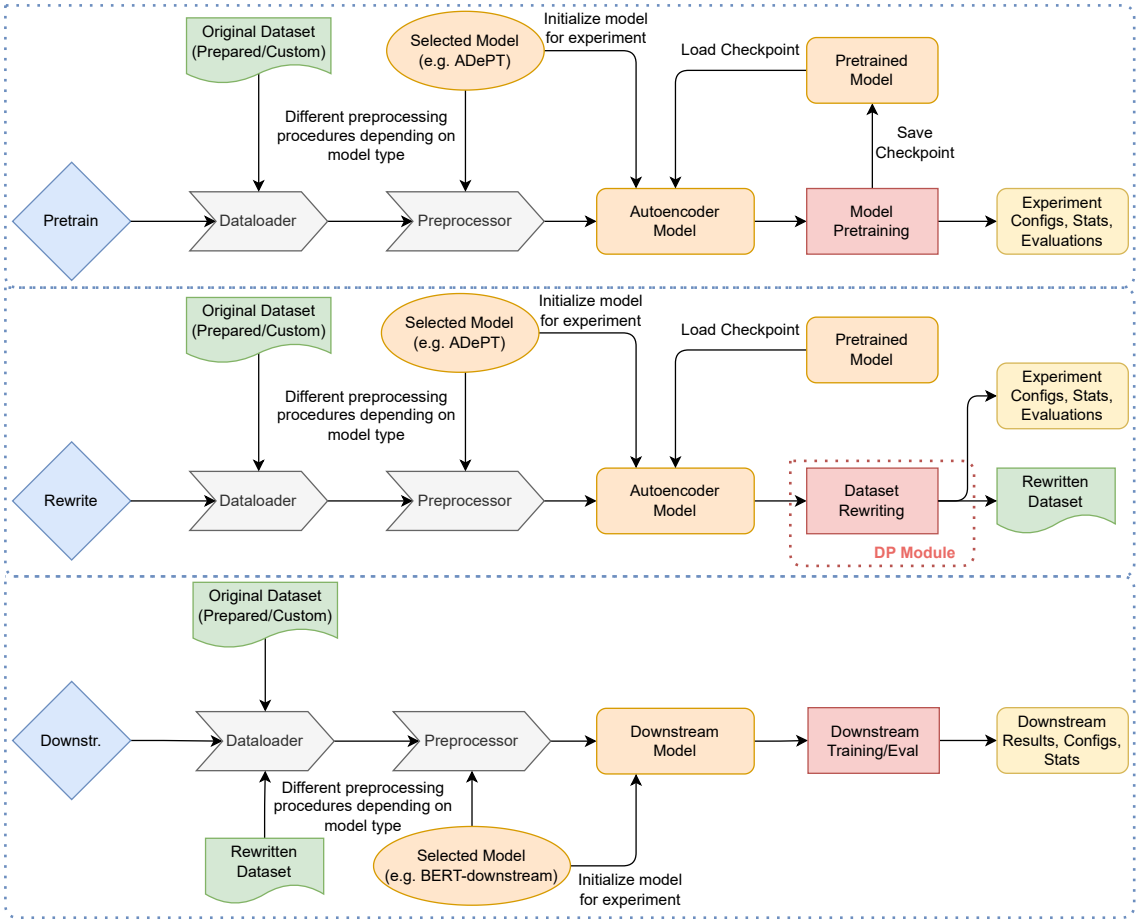


Figure 5.1: Overall structure of `DP-Rewrite`. Colors represent groupings of similar components. Blue: Experiment mode. Grey: Dataset preparation. Green: Datasets (original/rewritten). Orange: Model-related components. Red: Main experiment loop. Yellow: Additional experiment outputs.

The goal of our system is to provide a seamless way to perform differentially private text rewriting, both on existing and custom datasets. A user can either load a dataset that we provide out-of-the-box, or use a custom one. In addition, we want to make it fast and convenient to run experiments for existing methodologies in DP text rewriting (e.g. ADePT), as well as the ability to integrate novel approaches. For this, we have a general *autoencoder* class, based on which out-of-the-box and custom models are built. In this sense, our software is designed to be open and modular, where the researcher can swap out existing components to run a variety of experiments, as well as use the software for one's own privatized text rewriting needs.

The core architecture of our system can be seen in Figure 5.1. We divide the experiments into three distinct **modes**: *pre-training*, *rewriting*, and *downstream*. For all three, the pipeline begins with a dataloader which can either be a dataset provided in the framework, or a custom dataset specified by the user. Additionally, a rewritten dataset can be loaded for downstream experiments. The loaded dataset

is then preprocessed according to user-specified parameters and the user's selected model, split into different procedures depending on the model type (e.g. RNN-based, transformer-based). The model is then initialized, optionally from an existing checkpoint. At this point, the main experiment is run based on the specified mode, either (1) pre-training the autoencoder, (2) using an existing checkpoint to rewrite the dataset, or (3) running a downstream model on an original or rewritten dataset. For each mode, a variety of evaluations are available, such as BLEU (Papineni et al., 2002) and BERTScore (Zhang et al., 2019b) for pre-training and rewriting, and various classification metrics (e.g. F_1 score) for downstream experiments. The differential privacy component is incorporated during the rewriting phase for systems such as ADePT, although our framework also allows to incorporate it during the pre-training stage.

5.4 Case Study

We present here a case study that demonstrates the process of using our framework and provides insights into the ADePT system, for which we provide an implementation in the software. Our goal is to investigate the difference in rewritten texts and downstream evaluations when we pre-train an autoencoder on one dataset and use this to rewrite another dataset. If we notice a lot of tokens from the dataset used for pre-training in the rewritten dataset, as well as comparatively higher downstream scores when pre-training and rewriting on the same dataset, then we can be certain of another form of privacy leakage in ADePT. We first describe the ADePT model, including its model architecture and differential privacy module.

5.4.1 ADePT

ADePT starts out with a standard autoencoder architecture. Given an input document x, an encoder function Enc calculates a latent vector representation z. This representation is then sent to a decoder function Dec, which reconstructs the original text $\hat{y}$. ADePT uses a single-layer, unidirectional LSTM for both the encoder and decoder.

$$z = \textsc{Enc}(x) \quad \text{and} \quad \hat{y} = \textsc{Dec}(z) \tag{5.1}$$

To incorporate differential privacy into this model, the unbounded latent vector $z \in \mathbb{R}^n$ (where n is the size of the autoencoder's hidden dimension) is bounded by its norm and the clipping constant $C \in \mathbb{R}$. Laplace or Gaussian noise (η) is then added to the resulting vector, from which the decoder reconstructs the original sequence, $\hat{y}$. This is shown in Eqn 5.2.

$$z' = z \cdot \min\left(1, \frac{C}{||z||_2}\right) + \eta \tag{5.2}$$

For the experiments below, we make an adjustment to this system, fixing a theoretical issue in the sensitivity calculation of Eqn. 5.2, outlined in Habernal (2021). While in Krishna et al. (2021) the ℓ_1-sensitivity for the Laplace mechanism

is calculated as $\Delta_1 = 2C$, this is only the case if Eqn. 5.2 is modified to clip using the ℓ_1 norm instead, as in Eqn. 5.3.

$$z' = z \cdot \min\left(1, \frac{C}{||z||_1}\right) + \eta \tag{5.3}$$

Alternatively, it is possible to retain the ℓ_2 norm clipping procedure, in which case the ℓ_1 sensitivity has to be modified to $2C\sqrt{n}$, instead of $2C$. Here we opt for the former option, clipping using the ℓ_1 norm.

5.4.2 Datasets

As in Krishna et al. (2021), we use the ATIS (Dahl et al., 1994) and Snips (Coucke et al., 2018) datasets to conduct experiments on an intent classification task in English. For both datasets, we use the same train/validation/test split provided by Goo et al. (2018), with 4,478 training, 500 validation and 893 test examples for ATIS, and 13,084 training, 700 validation and 700 test examples for Snips. There are a total of 26 intent labels in ATIS and 7 in Snips.

5.4.3 Implementation

We start our experimental pipeline by pre-training two autoencoder models, one on ATIS (1) and the other on Snips (2), in both cases using the training split. For pre-training, we set the vocabulary to the maximum number of words in the training set. As in ADePT, we do not incorporate a differential privacy component during pre-training, although we clip encoder hidden representations with a clipping constant value of 5. We limit sequence lengths to a maximum of 20 tokens, pre-training for 200 epochs with a learning rate of 0.003.

We then use these two models for rewriting, applying both pre-trained models for rewriting the training and validation partitions of ATIS and Snips, resulting in four rewriting settings in total. For each setting, we rewrite using five ε values: ∞, 1000, 100, 10, and 1. We use the same clipping constant value of 5 as in pre-training.

Downstream experiment setup

For downstream experiments, we use a pre-trained `bert-base` model (Devlin et al., 2019),[1] with an additional feedforward layer that takes the mean of the last hidden states as input and predicts the output label. We use the rewritten training and validation sets for each configuration, and the original test sets for final evaluation. We run each configuration with five different random seeds and report the mean and standard deviation.

5.4.4 Results and Analysis

Our results can be seen in Figure 5.2, and in the complimentary Table 5.1. We observe the main patterns as follows. First and most importantly, datasets rewritten using a model that was pre-trained on the same dataset generally show better

[1] Available from `https://huggingface.co/bert-base-uncased`.

downstream results than datasets rewritten using a model pre-trained on a different dataset. For instance, at $\varepsilon = 1000$, rewritten Snips from a model pre-trained on Snips has an F_1 score of 0.94, while rewritten Snips from a model pre-trained on ATIS has only 0.20. In fact, this is true even at $\varepsilon = \infty$ (non-private setting), without any added noise (e.g. 0.94 F_1 pre-trained Snips, rewritten Snips vs. 0.18 F_1 pre-trained ATIS, rewritten Snips), since for the latter case the model ends up rewriting the dataset that it was pre-trained on, having memorized many of its examples. This can be seen in Figure 5.3, where the rewritten sentences appear to have no resemblance to the original dataset used for rewriting, but are very similar to the data used for pre-training.

Next, as expected, the results decrease for all configurations as the privacy budget ε decreases, except for rewritten ATIS from a model pre-trained on Snips, where results are low for all ε values, probably due to the same reasons as shown in Figure 5.3. At the lower ε values of 10 and 1, performance is very low for all configurations. Since there is so much noise added to the encoder hidden representations z, the utility of ADePT's rewriting is severely diminished, for any data inputs.

Finally, compared to running downstream experiments on the original dataset, Snips rewritten with a model pre-trained on Snips shows about the same results at high ε values (e.g. 0.94 F_1 pre-trained Snips, rewritten Snips vs. 0.95 F_1 original Snips for $\varepsilon = \infty$). ATIS rewritten with a model pre-trained on ATIS shows lower results in this case (e.g. 0.73 F_1 pre-trained ATIS, rewritten ATIS vs. 0.87 F_1 original ATIS for $\varepsilon = \infty$). We speculate that since ATIS is a smaller dataset, there are less data points to effectively pre-train ADePT for the autoencoding task. We additionally report random and majority baselines in Table 5.1 on the original datasets for comparison, which are both very low, likely due to the large number of classes in both datasets, especially ATIS.

We have thus shown that, despite fixing the theoretical privacy guarantees of ADePT outlined in Section 5.4.1, the pre-training procedure still results in privacy leakage, with rewritten datasets exposing a lot of information from the dataset used for pre-training. As a result, downstream performance is inflated if the datasets for pre-training and rewriting are the same.

5.5 Chapter Summary

In this chapter, we have begun our investigation into the privatization of textual data in the setting of local differential privacy. Before moving on to designing a differentially private text rewriting system that pushes the boundaries on the privacy/utility trade-off and improves upon previous systems, we have addressed a major concern that has been present in the field, namely the problem of transparency and reproducibility in current research on differentially private text rewriting.

We introduced our open-source LDP text rewriting framework, `DP-Rewrite`. We have demonstrated a sample use-case for our framework, which allows us to detect privacy leakage in the pre-training procedure of the ADePT system, an example of

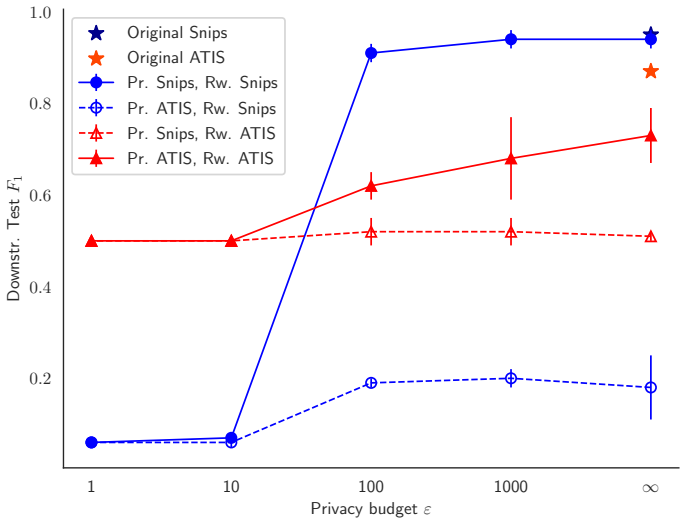


Figure 5.2: Downstream macro-averaged F_1 results for case study experiments with pre-trained and rewritten Snips/ATIS datasets, as well as comparisons with results on the original datasets ("Original Snips" and "Original ATIS"). Lower ε corresponds to better privacy.

Snips rewritten from ATIS $\varepsilon = 1000$	
Original Snips doc.	listen to westbam alumb allergic on google music
Rewritten Snips doc.	how many people fly on after a turboprop airport
ATIS doc. similar	how many people fly on a turboprop airport
ATIS rewritten from Snips $\varepsilon = 1000$	
Original ATIS doc.	what flights leave from phoenix
Rewritten ATIS doc.	start playing my disney spring
Snips doc. similar	start playing my disney playlist

Figure 5.3: Sample rewritten texts showing memorization by the ADePT model when pre-training and rewriting on different datasets. For a given document in the original dataset ("Original Snips/ATIS doc."), its rewritten version by the model ("Rewritten Snips/ATIS doc.") has no resemblance to it, but is very similar to another document from the pre-trained dataset ("ATIS/Snips doc. similar").

how the modular and customizable nature of the software allows for transparency and reproducibility in DP text rewriting research. `DP-Rewrite` is continuing to be under active development, and we are incorporating new datasets and private text rewriting methodologies as they are released. We welcome feedback from the community.

Pretr. Dat.	**Rewr. Dat.**	ε	**Test** F_1
Snips	Snips	∞	0.94 (0.02)
Snips	Snips	1,000	0.94 (0.02)
Snips	Snips	100	0.91 (0.02)
Snips	Snips	10	0.07 (0.01)
Snips	Snips	1	0.06 (0.00)
ATIS	Snips	∞	0.18 (0.07)
ATIS	Snips	1,000	0.20 (0.02)
ATIS	Snips	100	0.19 (0.01)
ATIS	Snips	10	0.06 (0.01)
ATIS	Snips	1	0.06 (0.01)
Snips	ATIS	∞	0.51 (0.01)
Snips	ATIS	1,000	0.52 (0.03)
Snips	ATIS	100	0.52 (0.03)
Snips	ATIS	10	0.50 (0.01)
Snips	ATIS	1	0.50 (0.01)
ATIS	ATIS	∞	0.73 (0.06)
ATIS	ATIS	1,000	0.68 (0.09)
ATIS	ATIS	100	0.62 (0.03)
ATIS	ATIS	10	0.50 (0.01)
ATIS	ATIS	1	0.50 (0.01)
Snips Orig.			0.95 (0.01)
ATIS Orig.			0.87 (0.03)
Snips Rand.			0.14
ATIS Rand.			0.01
Snips Maj.			0.03
ATIS Maj.			0.13

Table 5.1: Downstream macro-averaged F_1 results for case study experiments with pre-trained and rewritten Snips/ATIS datasets. We additionally show results on the original datasets, as well as random and majority baselines. Test F_1 shown as "mean (standard deviation)" over five runs with different random seeds. Lower ε corresponds to better privacy.

Chapter 6

DP-BART for Privatized Text Rewriting under Local Differential Privacy

In this final chapter, we continue to tackle the third and last research question, RQ3, on the privatization of the textual data itself. As in the previous chapter, we are working in the local DP setting on the task of privatized text rewriting. Having addressed the issues of transparency and reproducibility in this topic in the previous chapter, we now move on to the second part of our research question, in which our goal is to design an effective model for LDP text rewriting, improving the privacy/utility trade-off over previous methods. As we discuss in Chapter 5, one major issue in achieving this is the *strict adjacency constraint*, requiring any two rewritten documents to be indistinguishable from one another, given an ε privacy budget. The main consequence of this issue is that a significant amount of perturbation needs to be introduced to the input text, in order to achieve this ε-DP privacy guarantee for any target document. We design an LDP text rewriting system, **DP-BART**, in order to tackle these problems and discuss in detail the *limitations* of the task of private text rewriting with LDP.

6.1 Introduction

In the original randomized response technique of Warner (1965), individual bits of personal information are privatized in the LDP setting, with respect to a given individual. The core idea in privatized text rewriting is the same, in which we take a data point associated with an individual, and perturb it with LDP guarantees. However, the nature of the input information that is to be privatized is significantly more complicated than that of individual bits, with an entire textual document treated as the 'object of privatization', i.e. a single private data point. We provide an example of a rewritten text in Chapter 5 for intent classification, which we repeat here for clarity. If the input document is "I would like to fly from Denver to Los Angeles this Thursday", the corresponding output document, rewritten with LDP

guarantees, could then be "Show me flights to cities in California this week". In this example, we have obscured various types of information in the original document, including the origin and destination of the flight, the weekday of the flight, the fact that the author of the document is intending to fly, as well as the general writing style of the text that is unique to that author. In other words, all tokens and their combinations are concealed with respect to the individual, providing that individual plausible deniability that the document in question is associated with them, as opposed to *any other document*. This is analogous to flipping a coin in randomized response, with the owner of the private *bit* having plausible deniability that the bit they are providing is in fact the one associated with their true private data point. Again, this is a consequence of working in the LDP setting, in which any two data points are considered neighboring, as opposed to the global DP setting, where *any two datasets* are neighboring. In this latter case, the contribution of any document within a particular dataset would then be obscured in relation to some analysis, but the DP guarantee does not extend outside of that scope. LDP is thus a *stronger form* of differential privacy with respect to the provided privacy guarantees.

Overall, the benefits of an LDP text rewriting system are immense, where the output privatized dataset can be used for any downstream analysis, in addition to not requiring a *trusted data aggregator* to add random noise. As in the above flight example, we also avoid the problem of having to manually determine what specific tokens in a document are private, applying LDP to the entire document. Nevertheless, there is a significant difficulty in creating such a system, with a lot of perturbation required to achieve any reasonable privacy guarantees, leading to poor downstream utility, as we outline above. This is in addition to the issues we outlined in Chapter 5, in which exisitng DP text rewriting systems may suffer from formal flaws in their methodology (Habernal, 2021), use older types of models (e.g. the single-layer LSTM of the ADePT system (Krishna et al., 2021)), as well as utilize high privacy budgets.

In order to address these issues, we propose a new private text rewriting system, **DP-BART**, that outperforms existing LDP methodologies. Our approach consists of several techniques that can be directly applied to a pre-trained BART model (Lewis et al., 2020a), outlined in more detail in Section 6.2, without having to design and train such a model from scratch. Despite being a large transformer architecture, it can easily be used for data privatization, not requiring many resources. Our methodology consists of a novel clipping method for the BART model's internal encoder representations, as well as a pruning mechanism and additional training approach that reduces the amount of DP noise that needs to be added to the data during the privatization process.

The contributions for this chapter can thus be defined as follows. First, we present the **DP-BART** model and its related methodologies. The goal is to obtain a better privacy/utility trade-off by reducing the required amount of DP noise that is introduced during the rewriting process. Our primary baseline for this task is the ADePT model, which we discussed in detail in the previous chapter.

Next, in order to investigate the privacy/utility trade-off of our proposed method-

ologies in detail, we run experiments on five unique datasets and evaluate the rewritten texts for each dataset on their associated downstream text classification tasks. These datasets all have varying sizes, ranging from *small* (between ~5,000 and 15,000 data points), to *medium* (~25,000 data points), *large* (over 100,000 data points), and *very large* (around 2M data points). This variation in size allows us to additionally investigate to what extent more data is beneficial in the LDP setting. The intuition here is that, despite a more noisy training setting, a downstream model may be able to determine the important patterns in the input data that are necessary to correctly solve the downstream task.

Finally, we thoroughly examine the feasibility of the LDP text rewriting setting in Section 6.7. This is primarily related to the high noise requirement described above due to the strict adjacency constraints. We investigate the trade-offs between the possible privacy guarantees that can realistically be achieved and the downstream performance of a rewritten dataset. We then further discuss what exactly is the object of privatization, the required computational resources for our methodology in comparison to other alternatives, as well as limitations of the approach as a whole and what can be done to mitigate them.

6.2 Related Work and Background

Overview of LDP text rewriting Applying differential privacy to neural network training and model publishing has converged to using a mainstream method, namely DP-SGD (Abadi et al., 2016a). However, the task of text privatization is still broadly unexplored, with many unanswered questions remaining, such as dealing with the unstructured nature of text and explainability of the privacy guarantees provided to textual data (Klymenko et al., 2022). There are only a few approaches that directly tackle this problem of LDP text rewriting, with the ADePT system of Krishna et al. (2021) being the primary baseline method. For further information on previous methods in the literature on privatized text rewriting, we refer to Section 5.2 of Chapter 5.

Overview of the BART model Here we outline the core elements of the BART model and its relation to the LDP text rewriting task. BART is a sequence-to-sequence Transformer architecture (Vaswani et al., 2017), acting as a denoising autoencoder. It combines the BERT-like bidirectional encoder (Devlin et al., 2019) with the GPT-like left-to-right autoregressive decoder (Radford et al., 2018). The base model contains 6 layers for the encoder and decoder, with cross-attention performed over the final encoder layer. BART is pre-trained through a number of noise transformations of the input document, including token masking, token deletion, and sentence permutation, optimizing a cross-entropy reconstruction loss. One strong benefit of BART for differentially private text rewriting is that, by design, it is well-equipped for the autoencoding task of reconstructing corrupted documents.

Overview of pruning for neural networks There have been a variety of pruning methods derived for neural networks since the 1990s. The more popular tech-

nique is weight pruning (e.g. LeCun et al. (1989), Hassibi et al. (1993), Frankle and Carbin (2018)), reducing the size of a model and its computation time, but minimizing any negative impact to its performance. More distinct is structured pruning, such as neuron pruning (e.g. Kruschke and Movellan (1991), where the architecture of a network is reduced by eliminating full structures of the network, which is more in line with our approach. Regardless of the specific method used, a very common pipeline is to iteratively prune and further train a model, to help it recover from potentially lost performance (Han et al., 2015). Overall, pruning tends to be highly effective, with substantial compression possible for models (Blalock et al., 2020).

In contrast to the above goals of size and computational efficiency, we use pruning with the primary objective of *dimensionality reduction* on a specific hidden layer. This dimensionality is directly related to privacy concerns, with a lower dimension resulting in less added noise to the model, which allows us to use lower privacy budgets while maintaining better performance. Our neuron-based pruning approach is outlined in Section 6.3.4.

6.3 Methods

We outline this section as follows. First, we briefly recapitulate the main aspects of our modified version of the ADePT system, which we use as a baseline. Next, we investigate two main issues with applying a local DP system such as ADePT to a transformer model, namely extreme sensitivity and computational infeasibility, described in Section 6.3.2.

We then demonstrate several novel mechanisms which tackle these issues and provide numerous benefits in the privacy/utility trade-off for the local DP setting. Section 6.3.3 describes the clipping by value module, with an additional analysis at the end of the section on determining optimal settings for it. Sections 6.3.4 and 6.3.5 then describe the neuron-based pruning methods which significantly reduce the amount of noise that needs to be added to the model for a given privacy budget and increase model robustness to noise through further noisy training. Low-level specifics on the pruning methods are further provided at the end of Section 6.3.4.

6.3.1 Baseline (ADePT with adjustments)

Recall that ADePT is an LSTM-based autoencoder model, with encoder and decoder functions Enc and Dec, respectively, that take an input text document x and reconstruct it as $\hat{y}$, shown in Eqn. 5.1. The original ADePT model takes the latent encoder output $z \in \mathbb{R}^n$ representation and clips it by its ℓ_2 norm, with the clipping constant $C \in \mathbb{R}$, adding Laplace or Gaussian noise to the output based on the ℓ_2 sensitivity, as in Eqn. 5.2. For comparison with our primary methodologies below, we refer to this as the *clipping by norm* module.

In our experiments for the current chapter, we make three modifications to the original ADePT model. First, we again fix the theoretical issue in the sensitivity calculation of Eqn. 5.2. Instead of using the sensitivity of $2C$ for the Laplace noise

scale, outlined in Theorem 1 of Krishna et al. (2021), we instead use the corrected sensitivity of $2C\sqrt{n}$ from Theorem 5.1 of Habernal (2021). Second, the classical Gaussian mechanism guarantees privacy only for $\varepsilon < 1$ (Dwork and Roth, 2013, p. 262). We therefore utilize the Analytic Gaussian mechanism (Balle and Wang, 2018) instead, which allows us to use $\varepsilon \geq 1$.

Finally, we fix the issue with the pre-training procedure of the model. As we describe in Chapter 5, the original ADePT was pre-trained on the downstream datasets with clipping, but without the added DP noise from Eqn. 5.2. We saw that this results in significant memorization by the model of the input documents, even after adding DP noise during the rewriting process. In order to remedy this, we therefore pre-train the autoencoder model on a public corpus, unrelated to the downstream datasets.

6.3.2 Applying LDP to Transformers

There are two main issues in applying a transformer model to a local DP setting similar to ADePT, outlined below.

Using LDP in pre-trained transformers suffers from extreme sensitivity

First, we need a significantly larger amount of noise to be added to the model, due to the increased size of the encoder output vector. Due to the cross-attention mechanism typical of transformer models, the full output vector for the BART encoder is of size $d_{tok} \times l$, where d_{tok} is the hidden size for a particular token, while l is the sequence length. For the smaller `bart-base` model, using a short sequence length of 20, this results in a dimensionality of $768 \times 20 = 15360$. In comparison, ADePT's encoder output vector dimensionality is only 1024 in our configuration.

High requirement of computational resources for pre-training

We experimented with clipping by norm for BART, similarly to ADePT, but found that it destroys any useful representations of the model (even prior to adding the DP noise). Additional pre-training of BART that would incorporate clipping by norm turned out to be ineffective.

The remaining option to learn a model with clipping by norm would be to pre-train the model from scratch. Unlike the small ADePT model, which is a uni-directional, single-layer LSTM, pre-training a BART transformer from scratch is computationally infeasible on an academic budget. While the details of BART's computational requirements are not described in Lewis et al. (2020a), we can estimate this for the relatively small `bart-base` model of 139M parameters that was released by the original authors,[1] by comparison with other similar-sized models. For instance, the BERT model (Devlin et al., 2019), with less parameters (110M for `bert-base`), was pre-trained for 4 days on up to 16 TPUs, as described on the authors' Github repository.[2]

[1] https://github.com/facebookresearch/fairseq/tree/main/examples/bart
[2] https://github.com/google-research/bert

6.3.3 DP-BART-CLV (Clipping by Value)

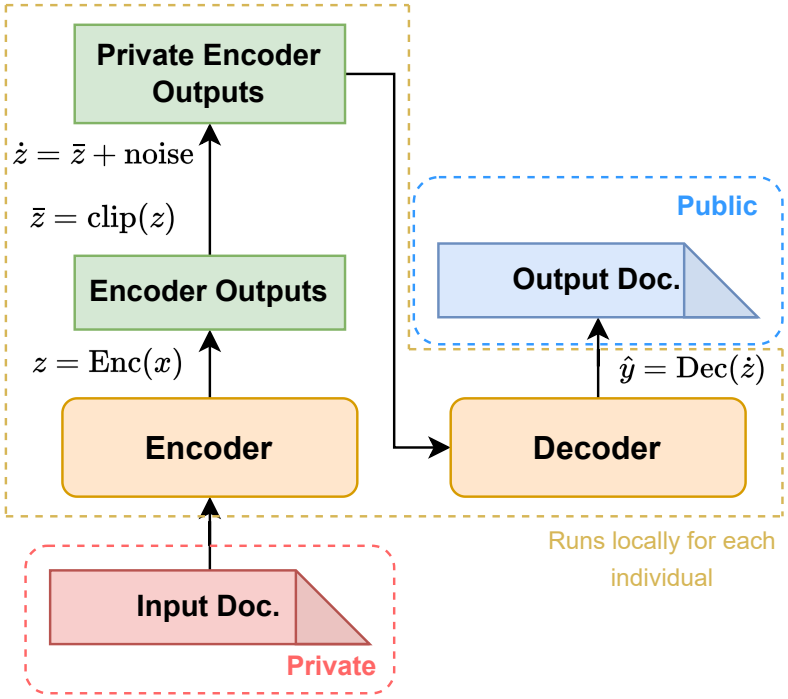


Figure 6.1: DP-BART-CLV model.

To address the issues with clipping by norm, we developed the **DP-BART-CLV** model, shown in Figure 6.1. We analyzed the internal representations of a pre-trained BART model's encoder output vector values, using a public dataset. We found that these are mostly bounded within a couple of standard deviations from their mean. We present this analysis in detail at the end of this section.

To avoid significantly altering these representations, we can therefore use clipping by value (CLV), as in Eqn. 6.1.

$$\bar{z}_i = \min(\max(z_i, C_{min}), C_{max}) \tag{6.1}$$

for any dimension i in the encoder output vector z, a set minimum threshold C_{min} and maximum threshold C_{max}. The bulk of values centered around the mean of z are thus left the same, without being rescaled as in Eqn. 5.2. Since these values were also found to be symmetrically distributed, we modify Eqn. 6.1 to set $C = C_{max} = -C_{min}$, as in Eqn. 6.2.

$$\bar{z}_i = \min(\max(z_i, -C), C) \tag{6.2}$$

The pipeline for **DP-BART-CLV** is as follows. We first initialize a BART model using a pre-trained checkpoint, where pre-training was again done on a public dataset, separate from the downstream datasets that are to be privatized.

For a given document, we put it through the encoder of the model at inference time, obtaining the encoder output vector z, as in Eqn. 6.3.

$$z = \text{Enc}(x) \tag{6.3}$$

where x is the input sequence and Enc is the encoder of the BART model. While the BART model outputs the encoder's last hidden state as $z \in \mathbb{R}^{l \times d_{tok}}$ for each mini-batch, we flatten this vector to be $z \in \mathbb{R}^n$, where $n = l \cdot d_{tok}$. Clipping is then performed as in Eqn. 6.4,

$$\bar{z} = \text{clip}(z) \tag{6.4}$$

where clip is carried out for every dimension of the vector, according to Eqn. 6.2.

With this clipping mechanism in place, we can now calculate its sensitivity, in order to determine the scale of noise to add in the DP setting. This is outlined in Theorems 6.3.1 and 6.3.2 below.

Theorem 6.3.1. *Let $f : \mathbb{R}^n \to \mathbb{R}^n$ be a function as in Eqn. 6.4. The ℓ_1 sensitivity $\Delta_1 f$ of this function is calculated as in Eqn. 6.5, where $C \in \mathbb{R} : C > 0$ is the clipping constant and $n \in \mathbb{N}$ is the dimensionality of the vector.*

$$\Delta_1 f = 2Cn \tag{6.5}$$

Proof. The ℓ_1 sensitivity of a function $f : \mathbb{R}^n \to \mathbb{R}^n$ is defined as: $\Delta_1 f = \max_{x,y} ||f(x) - f(y)||_1$, where $||x - y||_1 = 1$. Since in our case f clips every value to be in the range $[-C, C]$, the following inequality must be true.

$$\begin{aligned} ||f(x) - f(y)||_1 &= |f(x_1) - f(y_1)| + \dots \\ &\quad + |f(x_n) - f(y_n)| \\ &\leq |C - (-C)| + \dots \\ &\quad + |C - (-C)| \\ &= |2C| + \dots + |2C| \\ &= 2Cn \end{aligned} \tag{6.6}$$

This inequality also holds true when the C values are reversed for any summand, due to the absolute value: $|C - (-C)| = |-C - C|$. □

Theorem 6.3.2. *Let $f : \mathbb{R}^n \to \mathbb{R}^n$ be a function as in Eqn. 6.4. The ℓ_2 sensitivity $\Delta_2 f$ of this function is calculated as in Eqn. 6.7, where $C \in \mathbb{R} : C > 0$ is the clipping constant and $n \in \mathbb{N}$ is the dimensionality of the vector.*

$$\Delta_2 f = 2C\sqrt{n} \tag{6.7}$$

Proof. The ℓ_2 sensitivity of a function $f : \mathbb{R}^n \to \mathbb{R}^n$ is defined as: $\Delta_2 f = \max_{x,y} ||f(x) - f(y)||_2$, where $||x - y||_1 = 1$. As for the ℓ_1 sensitivity above, f clips every value to be in the range $[-C, C]$, so the following inequality must be true.

$$
\begin{aligned}
||f(x) - f(y)||_2 &= \sqrt{\begin{matrix}(f(x_1) - f(y_1))^2 + \dots \\ + (f(x_n) - f(y_n))^2\end{matrix}} \\
&\leq \sqrt{\begin{matrix}(C - (-C))^2 + \dots \\ + (C - (-C))^2\end{matrix}} \\
&= \sqrt{(2C)^2 + \dots + (2C)^2} \\
&= 2C\sqrt{n}
\end{aligned} \tag{6.8}
$$

This inequality also holds true when we reverse the position of the C values for any summand, $(C - (-C))^2 = (-C - C)^2$. □

We then add noise to this clipped vector, as in Eqn. 6.9.

$$\dot{z} = \bar{z} + (Y_1, \dots, Y_n) \tag{6.9}$$

where each Y_i is drawn i.i.d. from $\text{Lap}(\frac{\Delta_1}{\varepsilon})$ for the Laplace mechanism (Dwork and Roth, 2013) or $\mathcal{N}(0, (\frac{\alpha\Delta_2}{\sqrt{2\varepsilon}})^2)$ for the Analytic Gaussian mechanism, where α is calculated according to Algorithm 1 of Balle and Wang (2018).

Decoding is then performed auto-regressively (e.g. using beam search) as usual, using this perturbed $\dot{z}$ encoder output vector, instead of the original z vector, as in Eqn. 6.10.

$$\hat{y} = \textsc{Dec}(\dot{z}) \tag{6.10}$$

where $\hat{y}$ is the model's output prediction of the reconstructed input sequence x. By standard arguments, the **DP-BART-CLV** model satisfies $(\varepsilon, 0)$-DP for the Laplace mechanism and (ε, δ)-DP for the Analytic Gaussian mechanism, as outlined in Eqn. 6.9 (Dwork and Roth, 2013; Balle and Wang, 2018).

Selecting Clipping Value for DP-BART

When clipping encoder outputs by value for the DP-BART model, we want to choose left and right values C_{min} and C_{max} that capture the most information from the original vector. One way to go about this is to estimate the distribution of the encoder output vectors $z \in \mathbb{R}^n$ (see Eqn. 6.3) of a pre-trained BART model checkpoint, given several documents from an external public dataset, and then clip a certain number of standard deviations from the estimated mean. Performing an exploratory data analysis on these encoder output vectors, we noticed that they fairly closely match a Gaussian distribution, although with far more outliers.

In order to look into this more closely, we can perform Maximum Likelihood Estimation (MLE) to estimate the μ and σ^2 parameters, assuming the data follows a Gaussian distribution. For Gaussians, the MLE of these two parameters is simply the mean and variance of the existing data, respectively, in our case of the values of z, given an input document x. Based on multiple sample documents, we find that $\mu \approx 0.00$ and $\sigma \approx 0.2$. Hence, using the *66-95-99.7 rule* for normal distributions, we

can clip two standard deviations to the left and right, retaining 95% of the original values.

We therefore initially set $C = C_{max} = -C_{min}$, where $C = \mu + \sigma \cdot 2 = 0 + 0.2 \cdot 2 = 0.4$. Since μ is found to be 0, we are able to simplify the calculation to only have the C and σ parameters. In practice, we found that clipping only half of one standard deviation was enough to retain good performance, despite clipping away more information than what we estimate above. Hence, we set $C = \sigma/2 = 0.1$.

6.3.4 DP-BART-PR

We develop the **DP-BART-PR** model in order to address the remaining issue of dimensionality, outlined in Section 6.3.2. The **DP-BART-CLV** model, while being resource-efficient, still has the issue of a large dimensionality for the encoder output vectors, since in Eqns. 6.5 and 6.7, the sensitivity is multiplied by a factor of n and $\sqrt{n}$, respectively, which in turn results in a larger noise scale.

DP-BART-PR, addressing both the resource and dimensionality issues, is an extension to the above **DP-BART-CLV**, with an additional iterative pruning/-training mechanism applied to it. The procedure is outlined in Figure 6.2 and Algorithm 1.

As for **DP-BART-CLV**, we first load a pre-trained BART model checkpoint. Each input token will have an encoder output representation of dimensionality d_{tok}. For every token in the sequence, we prune a certain percentage of these neurons by setting them to 0. Importantly, *these pruned neurons are the same for every single input document*. The criteria for selecting these pruned neurons is discussed in more detail at the end of this section.

Following this pruning step, we train the model for k iterations to compensate for possible lost performance from pruning. This step is performed on an external public dataset, unrelated to any downstream texts that are to be privatized. During this process, we also clip each dimension of the BART encoder output vector z_i according to Eqn. 6.2, to encourage representations to be constrained within the ranges $-C$ and C to reduce potential negative performance impacts of clipping during the rewriting phase.

We note that only a few data points are necessary for this additional training step, maintaining the low-resource setting, outlined in Section 6.4.3. We then continue this two-step process iteratively, until a desired dimensionality reduction of the encoder output veector is reached. At the end of this process, the resulting model weights are frozen and the final pruned indices of the encoder output vector z are saved. The model is then used for text rewriting at inference time, just like in **DP-BART-CLV**, but with the additional pruning step, using the saved indices.

As a result of this process, we can significantly reduce n in Eqns. 6.5 and 6.7, which in turn reduces the resulting noise scale used in Eqn. 6.9. With less noise added to the encoder output vectors for any given ε value, we can thus expect a

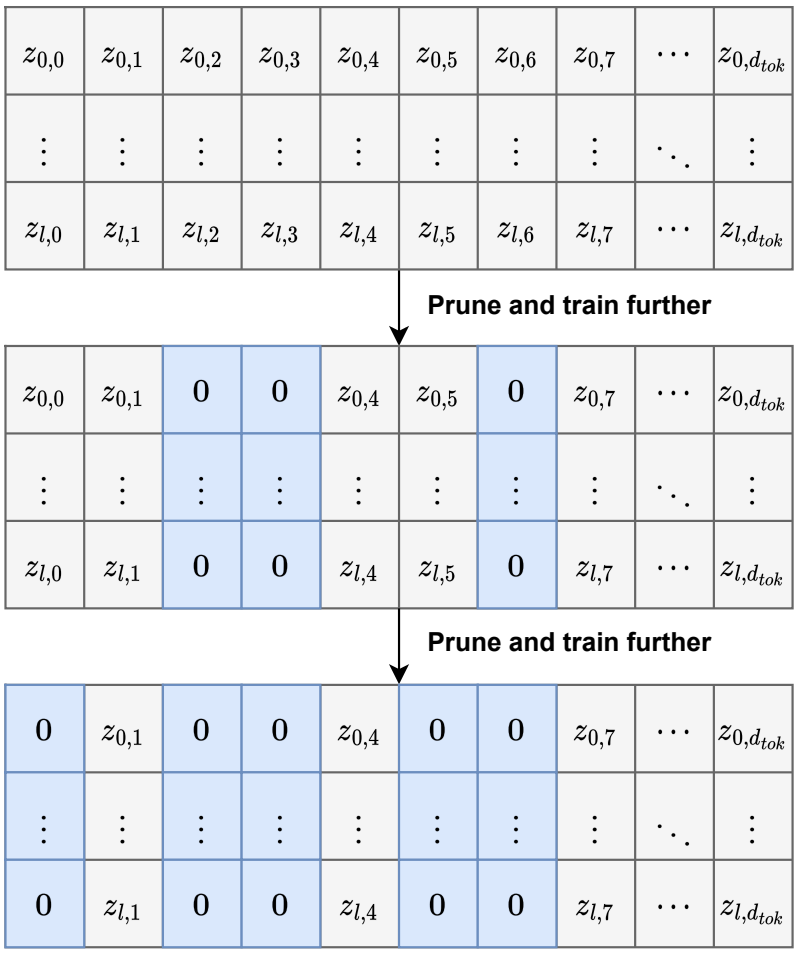


Figure 6.2: Pruning and re-training procedure for the DP-BART-PR model, illustrated for one document. Each i^{th} neuron from a set of indices is set to 0 for all tokens of the encoder output vectors $z \in \mathbb{R}^{l \times d_{tok}}$. These neuron indices are the same for any document. This process is repeated iteratively until performance starts to degrade.

better privacy/utility trade-off.

This pruning procedure can be seen as a *privacy/utility tuning knob*. With more pruning, we reduce the size of n, therefore requiring less added noise for a given ε value in the DP setting. At the same time, more pruning reduces the model's expressivity with less dimensions, which will result in an inevitable performance drop after reaching a certain pruning threshold. We noticed that pruning a few dimensions (e.g. 25% of neurons) can recover basically all of the performance of the model with some additional training steps, but after a certain point this starts to degrade. The 'sweet spot' we found is at approximately 75% of neurons. Additional discussion on these points can be found below in Section 6.3.4. We would like to stress again that these pruning adjustments are made just once and using public data only, after which the final model can be used locally by any individual for their own data privatization.

Algorithm 1 DP-BART Pruning

Input: Encoder: Enc_{θ_0}, Decoder: Dec_{θ_0}, Public dataset: $\mathcal{D}$, Encoder output dimension per token: d_{tok}, Number of epochs to additionally train: E

Output: Pruned model: Enc_{θ_E}, Dec_{θ_E}; Array of neuron indices to prune P of size d_{tok}

function PRUNE(z, P)
  ▷ $z \in \mathbb{R}^{l \times d_{tok}}$, where l is the seq. length
  for j in 1 to d_{tok} **do**
    if j in P **then**
      ▷ Set that neuron to 0 for all tokens
      $z[:, j] \leftarrow 0$
  return z
function ITER_PR($\mathcal{D}$, Enc_θ, Dec_θ, P)
  ▷ Iterate with pruning
  for each document x in $\mathcal{D}$ **do**
    Compute encoder outputs, $z \leftarrow \text{Enc}_\theta(x)$
    Prune, $z_{pr} \leftarrow \text{PRUNE}(z, P)$
    Decode, $\hat{y} \leftarrow \text{Dec}_\theta(z_{pr})$
    Compute loss on $\hat{y}$ and optimize
function ADD_P_IDXS(P)
  $new\_idxs \leftarrow$ select k values in $[1, d_{tok})$
  Append new_idxs to P
  return P

$P \leftarrow$ new Array
for epoch e in 1 to E **do**
  $P \leftarrow \text{ADD\_P\_IDXS}(P)$
  $\text{ITER\_PR}(\mathcal{D}, \text{Enc}_{\theta_e}, \text{Dec}_{\theta_e}, P)$
return Enc_{θ_E}, Dec_{θ_E}, P

Selecting Neurons for Pruning

At each pruning/training iteration for preparing the **DP-BART-PR** model, we need some criteria for selecting the next set of neuron indices that will be set to 0. Our method for selecting these is generally in line with previous work on pruning (Blalock et al., 2020), using weight magnitudes to determine relative importance of those weights.

As we discuss in Section 2.3.3 of Chapter 2, in the cross-attention module of the decoder of a transformer model such as BART, there are three initial projections of the input or target representations: Key (K), Query (Q), and Value (V). The K and V projections come directly from the encoder output vectors multiplied by a weight matrix for each, while the Q projection comes from the decoder's intermediate representations multiplied by a weight matrix. We can therefore choose the weight matrix of either the K or V projection from the cross-attention module of one of the decoder's layers. For this weight matrix, we take the sum of absolute values

of all weights associated with a particular neuron, to give a general indication of its importance. Given the distribution of these values associated with each neuron, we take the 25% quantile as the threshold. Any neuron with a value below this threshold is selected for pruning and set to 0.

At the next pruning iteration, after further training, we repeat the above process, this time only taking into account neurons that have not already been set to 0. We again calculate each neuron's relative importance value, taking the 25% quantile of these new values as the next threshold, and selecting any neurons with an associated importance value below it for pruning.

We found that taking the weight matrix of the K projection from the initial decoder layer outperformed all other configurations, such as using subsequent layers, or the V projection. Additionally, we found the above method to outperform randomly pruning neurons.

We perform two additional tweaks to this process to improve results further. First, we include the clipping by value procedure, with $C = 0.2$, when further training the model at each pruning iteration. We found that, without this step, the encoder output representations tend to shift to a distribution of values with a greater standard deviation. This then requires a larger C value when determining the mechanism's sensitivity in Eqns. 6.5 and 6.7, which in turn requires a greater noise scale in Eqn. 6.9. By including this clipping, we encourage encoder output representations to continue to primarily stay within the range $(-C, C)$.

The other tweak that we found to further improve results is to prune and further train the BART model for k iterations, but then use the neuron indices for pruning from the $k-1$ iteration. Performing this full pruning pipeline on a public dataset, we found that the best BLEU scores for rewriting at various ε values are after pruning/training the model for 6 iterations, then using the pruning indices from the 5^{th} iteration for actual rewriting of downstream datasets. This amounts to a total of 586 out of 768 (76.30%) neurons pruned for each token.

In theory, this pruning procedure could be replaced with another dimensionality reduction technique for the last hidden state of the encoder outputs (e.g. a bottleneck layer and its inverse). In our experiments, however, the above pruning procedure produced suprior results when trying various options for such a bottleneck layer. This includes architectures such as a feedforward neural network and CNN (LeCun et al., 1998), as well as various training methods (e.g. training these layers separately and reinserting them into the final full model, or training the full model together with these layers).

Proof that DP-BART-PR is differentially private

Theorem 6.3.3. *The **DP-BART-PR** model, combining Algorithm 1 and the above **DP-BART-CLV** procedure, summarized in Eqn. 6.9, satisfies $(\varepsilon, 0)$-DP when using the Laplace mechanism and (ε, δ)-DP when using the Analytic Gaussian mechanism.*

Proof. The procedure outlined in Algorithm 1 is performed on a public dataset, unrelated to the downstream data that is considered sensitive, hence no privacy budget is used up.

The remaining rewriting procedure with the pruned indices is exactly the same as for **DP-BART-CLV**, just at a lower dimension. The neuron indices that are set to 0 are the same for any input document. This means that no information from the input is encoded in these neuron indices. From the DP point of view, these zeroed neurons are the same for any two neighboring data points. Therefore, these neurons have no contribution to the DP sensitivity and do not require any privatization. The same proofs are therefore valid as for Theorems 6.3.1 and 6.3.2 for the Laplace and Analytic Gaussian mechanisms, respectively. □

6.3.5 DP-BART-PR+

We further augment the above **DP-BART-PR** model by incorporating additional training steps with added DP noise. This model follows the same procedure for iterative pruning and additional training, as outlined in Algorithm 1, but we add further training iterations on the pruned model with added DP noise to the clipped encoder output representations, as in Eqn. 6.9. For example, using the Analytic Gaussian mechanism at $\varepsilon = 500$, at each iteration we clip the encoder output vectors z from Eqn. 6.3 and add the appropriate amount of Gaussian noise based on the sensitivity from Eqn. 6.7.

The idea behind this additional training is to help the model to better decode from the noisified encoder representations. As with **DP-BART-PR**, for **DP-BART-PR+** we perform these additional training iterations on a public dataset, unrelated to the downstream datasets for privatized text rewriting. A separate model is prepared for each individual privacy budget ε.

6.4 Experiments

6.4.1 Datasets

We perform experiments on five English-language textual datasets, each gradually increasing in size (Table 6.1). For comparison with Krishna et al. (2021), we use ATIS (Dahl et al., 1994) and Snips (Coucke et al., 2018) as our *small* datasets, with the task of multi-class intent classification. We use the same train/validation/test split as in Goo et al. (2018), as in Chapter 5. For our *medium*-sized dataset, we use the popular IMDb dataset (Maas et al., 2011), on the binary classification task of movie review sentiment analysis. For this, as well as the following two datasets, we use a validation partition by randomly selecting 20% of the training set.

For a large dataset, we use the dataset from Gräßer et al. (2018), which is a collection of drug reviews from the website Drugs.com, also with the task of binary sentiment analysis, as in Shiju and He (2022). This dataset, although publicly available, closely simulates a sensitive dataset in need of privacy protection, with de-

tailed descriptions by users of their medical conditions and experiences with different treatments.

Our final dataset is the much larger Amazon Customer Reviews dataset (He and McAuley, 2016), of which we take a 2M subset of reviews from various categories (e.g. electronics, office products), from the full 144M. As with Drugs.com, we modify the original five-star sentiment score to a binary classification task, with four or more stars being the *positive* class, while the rest are *negative*.

We outline the last two datasets, Drugs.com and Amazon reviews, in more detail below.

Dataset	**Classes**	**# Trn.+Vld.**	**# Test**
ATIS	26	4,978	893
Snips	7	13,774	700
IMDb	2	25,000	25,000
Drugs.com	2	161,297	53,766
Amazon	2	1,904,197	211,605

Table 6.1: Dataset statistics. Trn.: Train, Vld.: Validation. Size represents number of documents.

Drugs.com reviews dataset

We present additional statistics on the Drugs.com dataset in Table 6.2. We note the class imbalance of the original dataset, where the majority class was the highest rating 9, from a score of 0 to 9, which accounted for approximately 17% of the total training set. This contributes to the relative imbalance of the positive and negative classes in our binary class version of the dataset.

	# Train	**# Test**
# Positive	97,410	32,349
# Negative	63,887	21,417
# Total	161,297	53,766

Table 6.2: Class distributions and total documents for the Drugs.com reviews dataset. Original classes 8 and 9 converted to the *positive* class, while the rest to the *negative* class for our experiments.

Amazon reviews dataset

For the Amazon dataset, since using the full 144M reviews is too computationally expensive, we reduce this to a more practical size, while still being comparatively larger than the other downstream datasets. To prepare a subset of the full Amazon dataset, we first select several product categories based on four criteria. (1) The category is large enough (e.g. $> 2M$ reviews). (2) Label 5 for the star rating is not too dominant (e.g. $< 60\%$), see general imbalance outlined in Table 6.3. (3) Label 4 for the star rating is also not too dominant (e.g. $< 60\%$), since we are merging

labels 5 and 4 into the *positive* class. (4) Label 1 for the star rating has enough representation (e.g. $> 10\%$).

We selected a total of 7 product categories, which matched at least three out of four of these criteria. From these reviews, we then filtered to include only those with 20 tokens or less, to fit our experimental scenario of shorter documents (outlined in more detail in Section 6.4.3). We then reduced this further by balancing positive and negative classes, with uniform probability selecting only N_{neg} positive label reviews, where N_{neg} is the number of negative labels in our current subset. Finally, we uniformly selected two-thirds of the resulting balanced dataset to reach the final size of approximately 2M reviews. We present each product category and its corresponding size in Table 6.3.

Importantly, we have a well-defined train-test split, taking 10% of the processed dataset and setting it aside for final downstream test evaluations. We release the specific document indices of our subset from the original large Amazon reviews dataset.[3] We present the final dataset statistics in Table 6.4.

Product Cat.	**# Docs. (original)**	**# Docs. (subset)**
Digital_Video_Games_v1_00	145,341	11,375
Electronics_v1_00	3,093,869	201,708
Lawn_and_Garden_v1_00	2,557,288	202,226
Major_Appliances_v1_00	96,901	4,940
Mobile_Apps_v1_00	5,033,376	536,550
Office_Products_v1_00	2,642,434	182,202
Wireless_v1_00	9,002,021	976,801
Total	22,571,320	2,115,802

Table 6.3: Product categories and corresponding number of documents from the full Amazon reviews dataset (mid), as well as from our prepared subset (right).

	# Train	**# Test**
# Positive	952,153	105,797
# Negative	952,044	105,808
# Total	1,904,197	211,605

Table 6.4: Final class distributions and total reviews for our Amazon reviews subset.

6.4.2 Experimental Setup

We have three main experimental configurations. The first is the **original** setting, where we run experiments on our downstream datasets without any rewriting or DP. The second configuration is **rewrite-no-dp**, where we utilize each of the four models outlined in Section 6.3 at $\varepsilon = \infty$ (**ADePT**, **DP-BART-CLV**, **DP-BART-PR**, and **DP-BART-PR+**). Finally, the third and main configuration is **rewrite-dp**,

[3] Original full dataset available on Huggingface at https://huggingface.co/datasets/amazon_us_reviews, our subset available at https://github.com/trusthlt/dp-bart-private-rewriting.

where we compare the above four models, this time at various privacy settings ($\varepsilon \in [10, 2500]$, Laplace and Analytic Gaussian mechanisms).

For **rewrite-no-dp** and **rewrite-dp**, our experimental pipeline consists of the following four steps, depending on the specific model used:

Pre-training: The model is pre-trained on a large public corpus. For **ADePT**, we use 50% of the Openwebtext corpus (Gokaslan and Cohen, 2019). For all our BART experiments, we load a pre-trained `facebook/bart-base` model.[4]

Further training: Only for **DP-BART-PR** and **DP-BART-PR+**, again performed using the Openwebtext corpus. It helps the model adjust to pruning and DP noise, respectively (as outlined in Sections 6.3.4 and 6.3.5). More details on the amount of further training are presented in Section 6.4.3.

Rewriting: We take a pre-trained model and rewrite one of the downstream datasets.

Downstream: We take the rewritten dataset (training and validation partitions) and run downstream experiments on it using a pre-trained BERT model with a classification head on top. We use the rewritten validation set for hyperparameter optimization (see Section 6.4.3) and the original test set for final evaluations. We present further details on the downstream model in Section 6.4.2.

In the **original** setting, we use the same downstream model as above, using the original datasets instead of the rewritten ones.

Evaluation We perform two types of evaluations for the above experimental settings: intrinsic and extrinsic. For our extrinsic evaluation we measure the test F_1 scores on the downstream task performance. This is the primary utility metric of the rewritten texts, with privacy correspondingly quantified with the ε value. We expect that even if a text may be rewritten to look very different from the original input, it could still have enough downstream task-specific information remaining to properly train a model on this task (e.g. the sentiment of a document in the case of sentiment analysis). This is in fact the 'sweet spot' we are looking for, removing identifying elements of the author, but still retaining some key features from the input for good downstream performance. We also measure BLEU scores (Papineni et al., 2002) for our intrinsic evaluation, discussed in more detail in Section 6.5.2.

Downstream Experimental Setup

As in our experiments for Chapter 5 (see Section 5.4.3), we use a pre-trained BERT model (Devlin et al., 2019) for running downstream experiments on the rewritten texts. We add a feedforward layer on top of the BERT model, taking as input the mean of its last hidden states. The model predicts the output label for text classification. For training the model and running validation, we use the rewritten training and validation partitions for each downstream dataset, at a given privacy

[4] Available from https://huggingface.co/facebook/bart-base.

configuration. For final evaluation, we run the model on the original test set of each dataset.

6.4.3 Hyperparameter configuration

For all our model configurations, we use a sequence length of 20 tokens. This limits the sensitivity in Eqns. 6.5 and 6.7 for our three BART models. For the ADePT model, we found that it is generally ineffective at the autoencoding task when using larger sequence lengths, presumably due to the problem of vanishing gradients for RNN-based models (Pascanu et al., 2013), as discussed in Section 2.3.3 of Chapter 2. Our search space for learning rates is in the range $[10^{-6}, 0.01]$. We use batch sizes of either 32 or 64.

When pre-training ADePT, we include the clipping procedure from Eqn. 5.2, otherwise the model is unable to properly rewrite a given input document, since the clipping significantly alters the encoder output representations. Additional hyperparameters for ADePT include an embedding size of 300 with pre-trained GloVe embeddings[5] (Pennington et al., 2014) and a hidden size of 512. Combining the LSTM cell and hidden state sizes, the ADePT encoder output vectors have a dimensionality of $512 \cdot 2 = 1024$.

For rewriting using the Analytic Gaussian mechanism, we always keep the δ value below $1/N$, where N is the total number of documents for a given dataset. As we discuss in Section 2.2.1 of Chapter 2, this is based on the idea that using a δ value that is overly large in relation to the dataset size can lead to potential privacy leaks, hence maintaining $\delta \ll 1/N$ is a good guideline to follow (Abadi et al., 2016b). We therefore use a δ value of 10^{-5} for the ATIS, Snips, and IMDb datasets, 10^{-6} for the Drugs.com dataset, and 10^{-7} for Amazon reviews. We perform rewriting with beam search, using a beam size of 10.

When performing additional training for the **DP-BART-PR** model, we again use the Openwebtext corpus. At each stage of pruning, we train the model for 500 iterations at a batch size of 32. In the case of further training for the **DP-BART-PR+** model, we again use the Openwebtext corpus, with the same number of iterations and batch size, but performed over multiple epochs. The number of epochs ranges from 100 to 500, for the different ε values from 2500 down to 10, based on the prediction loss and intermediate model outputs. We applied these further training steps to the **DP-BART-CLV** model as well to account for the potential effects of this training alone, but we did not find any improvements. This is in line with the high dimensionality issue of **DP-BART-CLV** destroying input representations in the private setting, which this additional training does not resolve without the pruning adjustments of the **DP-BART-PR(+)** models.

Regarding downstream text classification experiments, we run each configuration for a maximum of 50 epochs with three random seeds and report the mean. We use an early stopping patience of 5 epochs. We also report the standard deviation

[5] Downloaded from `https://nlp.stanford.edu/data/glove.6B.zip`.

in Section 6.5.2. We outline our choice of the clipping by value constant C in Section 6.3.3 and amount of pruning in Section 6.3.4.

Finally, our computational runtimes are under 1 hour for each configuration that does not use the Amazon dataset. The only exception to this is the Drugs.com reviews dataset, which reaches up to 2 hours 10 minutes for rewriting with the DP-BART models. The Amazon dataset takes significantly longer, with approximately 24 hours for rewriting with ADePT, 47 hours rewriting with DP-BART models, as well as up to 18 hours for downstream experiments, depending on when the early stopping condition is reached. We run experiments on a 32GB NVIDIA V100 Tensor Core GPU.

6.5 Results

6.5.1 Extrinsic evaluations

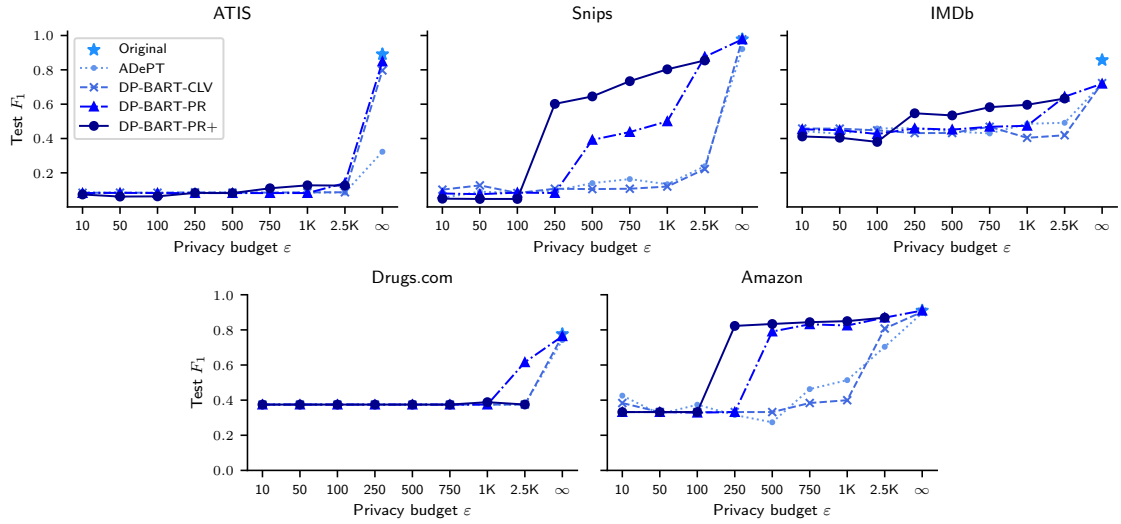


Figure 6.3: Downstream test F_1 results (macro-averaged) for each dataset, using the four model types. Lower ε corresponds to better privacy. Both **original** and **rewrite-no-dp** results can be seen on the right of each graph at $\varepsilon = \infty$. The rest of the results represent the **rewrite-dp** setting at different ε values.

Figure 6.3 shows our downstream test F_1 results for all datasets, at varying values of ε. We report results for the Analytic Gaussian mechanism, which nearly always outperformed those of the Laplace mechanism. We present results in tabular form with mean and standard deviations below in Section 6.5.2. Additionally, we present sample rewritten texts in Section 6.5.3. We outline the main patterns as follows.

DP-BART-PR+ performs best **DP-BART-PR+** reaches the best privacy/utility trade-off for the majority of datasets, having the highest scores at the lower ε values. **DP-BART-PR** results are second-best for most datasets, performing better than **DP-BART-CLV** and **ADePT**, which are low for the majority of configurations. The overall results hierarchy can be clearly seen in the Snips dataset,

where at $\varepsilon = 500$, **DP-BART-PR+** reaches F_1 0.65, **DP-BART-PR** at 0.39, while both **DP-BART-CLV** and **ADePT** are below F_1 0.15.

Original vs. rewritten Results for the **original** setting are generally on-par with those of the **rewrite-no-dp** setting. For instance, Snips original F_1 is 0.98, and $\varepsilon = \infty$ with rewriting is also at F_1 of 0.98 for **DP-BART-PR**, being very similar for the other three models. One exception to this is IMDb, which has a drop from original F_1 0.86 to 0.72 for all models. This can be explained by the fact that the **original** settings use longer sequence lengths, while both **rewrite-no-dp** and **rewrite-dp** settings are limited to a sequence length of 20. This is not a problem for datasets such as ATIS and Snips, since their documents are generally very short, mostly limited to brief user inquiries. For a dataset such as IMDb, however, which consists of detailed reviews by individuals, limiting the sequence length results in a loss of valuable information.

Epsilon vs. dataset size Regardless of dataset size, we can see a drop in results for all models as ε is decreased. With the models incorporating pruning, this drop appears at later ε values, such as **DP-BART-PR+** on the Amazon dataset moving down from F_1 0.82 at $\varepsilon = 250$ to F_1 0.33 at $\varepsilon = 100$, and **DP-BART-PR** from F_1 0.79 at $\varepsilon = 500$ to F_1 0.33 at $\varepsilon = 250$. A similar pattern can be seen for the Snips dataset, despite being far smaller than Amazon, while the Drugs.com dataset shows low results throughout, for all model types. The smallest dataset, ATIS, also performs poorly, which can be explained by the large number of classes and few data points for learning the task in the noisy setting. We can generally see that a larger dataset size does not necessarily mean better results at lower ε values, although the significantly larger Amazon dataset does show the best results.

6.5.2 Intrinsic evaluations

For intrinsic evaluations, we use BLEU scores to measure how close the input and rewritten output texts are to one another. Despite some criticisms of BLEU as a general-purpose evaluation metric for text generation (e.g. Callison-Burch et al. (2006)), it perfectly fits our scenario. Being a metric of n-gram overlap, it allows us to compare how similar the inputs and outputs are. In a way, a very high BLEU score points to privacy leakage, since it is showing how much of the original text remains in the output. We would therefore expect well privatized texts to have a relatively low BLEU score.

Our results can be seen in Table 6.5 for rewriting the training partition of each dataset with the Analytic Gaussian mechanism, together with the detailed downstream test F_1 results.

We can see that the BLEU scores for the training partition of each dataset show a largely positive correlation with the test F_1 downstream results, where a decrease in the former also indicates a decrease in the latter. For instance, the Snips dataset shows a BLEU score of 0.31 at $\varepsilon = 2500$ for **DP-BART-PR+**, with a test F_1 score of 85%. At $\varepsilon = 750$, this drops down to 0.23 BLEU score and 73% test F_1. By

$\varepsilon = 250$, the BLEU score is at 0.07, with the test F_1 score at 60%. Interestingly, despite lower BLEU scores, the downstream model is still able to sometimes learn the task successfully, obtaining a good F_1 score on the original test set.

Another example of this can be seen for the **DP-BART-PR** model on the Amazon dataset at $\varepsilon = 1000$, with a BLEU score of 0.17, reaching a test F_1 of 82%. A similar instance is **DP-BART-PR**+ rewriting Amazon at $\varepsilon = 250$, with a BLEU score of 0.15 and a test F_1 of 82%, compared to the non-private F_1 of 91%. This is in line with the goals of text privatization, where original identifying elements of the text are removed, but key features from the input are retained for good downstream performance.

Dataset	ε	Original	ADePT		DP-BART-CLV		DP-BART-PR		DP-BART-PR+	
		Test F_1	BLEU	Test F_1	BLEU	Test F_1	BLEU	Test F_1	BLEU	Test F_1
Snips	∞	0.98 (0.00)	6.34	0.92 (0.00)	98.41	0.98 (0.00)	54.39	0.98 (0.00)	N/A	N/A
	2,500		0.16	0.24 (0.13)	0.02	0.22 (0.14)	2.19	0.88 (0.03)	0.31	0.85 (0.02)
	1,000		0.03	0.13 (0.09)	0.00	0.12 (0.10)	0.07	0.50 (0.07)	0.30	0.80 (0.02)
	750		0.02	0.16 (0.08)	0.00	0.11 (0.07)	0.02	0.44 (0.11)	0.23	0.73 (0.04)
	500		0.01	0.14 (0.08)	0.00	0.11 (0.06)	0.01	0.39 (0.10)	0.22	0.65 (0.01)
	250		0.01	0.10 (0.09)	0.00	0.11 (0.07)	0.00	0.08 (0.02)	0.07	0.60 (0.03)
	100		0.01	0.08 (0.02)	0.00	0.08 (0.02)	0.00	0.08 (0.02)	0.00	0.05 (0.02)
	50		0.01	0.09 (0.05)	0.00	0.13 (0.06)	0.00	0.08 (0.03)	0.00	0.05 (0.02)
	10		0.01	0.05 (0.01)	0.00	0.10 (0.04)	0.00	0.08 (0.03)	0.00	0.05 (0.01)
ATIS	∞	0.89 (0.01)	16.04	0.32 (0.01)	97.45	0.80 (0.03)	69.26	0.85 (0.01)	N/A	N/A
	2,500		0.45	0.09 (0.00)	0.02	0.09 (0.00)	2.13	0.14 (0.07)	0.24	0.13 (0.07)
	1,000		0.06	0.09 (0.00)	0.01	0.09 (0.00)	0.06	0.08 (0.00)	0.25	0.13 (0.03)
	750		0.05	0.08 (0.00)	0.00	0.08 (0.00)	0.03	0.08 (0.00)	0.24	0.11 (0.05)
	500		0.03	0.09 (0.00)	0.00	0.08 (0.00)	0.01	0.08 (0.00)	0.11	0.08 (0.00)
	250		0.01	0.08 (0.00)	0.00	0.09 (0.00)	0.01	0.08 (0.00)	0.08	0.08 (0.00)
	100		0.01	0.08 (0.00)	0.00	0.08 (0.00)	0.00	0.08 (0.00)	0.00	0.06 (0.04)
	50		0.01	0.08 (0.00)	0.00	0.08 (0.00)	0.00	0.08 (0.00)	0.00	0.06 (0.04)
	10		0.01	0.09 (0.00)	0.00	0.08 (0.00)	0.00	0.08 (0.00)	0.00	0.07 (0.02)
IMDb	∞	0.86 (0.00)	95.00	0.72 (0.00)	93.49	0.72 (0.00)	89.05	0.72 (0.00)	N/A	N/A
	2,500		1.74	0.49 (0.04)	0.22	0.42 (0.04)	7.08	0.64 (0.02)	1.69	0.63 (0.01)
	1,000		0.18	0.49 (0.06)	0.16	0.40 (0.05)	0.25	0.47 (0.04)	1.04	0.60 (0.02)
	750		0.07	0.43 (0.08)	0.15	0.47 (0.03)	0.15	0.47 (0.05)	0.76	0.58 (0.02)
	500		0.04	0.44 (0.02)	0.12	0.43 (0.03)	0.05	0.45 (0.05)	0.52	0.53 (0.04)
	250		0.03	0.46 (0.02)	0.11	0.43 (0.02)	0.09	0.46 (0.02)	0.32	0.55 (0.03)
	100		0.02	0.46 (0.03)	0.08	0.45 (0.03)	0.06	0.43 (0.06)	0.00	0.38 (0.03)
	50		0.01	0.43 (0.01)	0.08	0.46 (0.08)	0.04	0.45 (0.05)	0.00	0.40 (0.06)
	10		0.01	0.44 (0.07)	0.05	0.46 (0.04)	0.03	0.45 (0.07)	0.00	0.41 (0.06)
Drugs.com	∞	0.78 (0.02)	92.41	0.74 (0.01)	93.46	0.77 (0.01)	88.47	0.76 (0.01)	N/A	N/A
	2,500		1.62	0.37 (0.00)	0.15	0.37 (0.00)	5.59	0.62 (0.02)	0.99	0.38 (0.00)
	1,000		0.12	0.37 (0.00)	0.08	0.37 (0.00)	0.15	0.37 (0.00)	0.46	0.39 (0.02)
	750		0.05	0.37 (0.00)	0.07	0.37 (0.00)	0.08	0.37 (0.00)	0.38	0.37 (0.00)
	500		0.03	0.37 (0.00)	0.06	0.37 (0.00)	0.05	0.37 (0.00)	0.28	0.37 (0.00)
	250		0.02	0.37 (0.00)	0.06	0.37 (0.00)	0.05	0.37 (0.00)	0.20	0.37 (0.00)
	100		0.01	0.37 (0.00)	0.05	0.37 (0.00)	0.04	0.37 (0.00)	0.00	0.37 (0.00)
	50		0.01	0.37 (0.00)	0.04	0.37 (0.00)	0.03	0.37 (0.00)	0.00	0.37 (0.00)
	10		0.01	0.37 (0.00)	0.04	0.37 (0.00)	0.03	0.37 (0.00)	0.00	0.37 (0.00)
Amazon	∞	0.91 (0.00)	26.96	0.90 (0.00)	96.52	0.90 (0.00)	57.16	0.91 (0.00)	N/A	N/A
	2,500		0.57	0.70 (0.01)	0.24	0.81 (0.04)	3.44	0.87 (0.01)	0.87	0.87 (0.00)
	1,000		0.09	0.51 (0.01)	0.22	0.40 (0.12)	0.17	0.82 (0.01)	0.66	0.85 (0.00)
	750		0.06	0.46 (0.15)	0.20	0.38 (0.09)	0.13	0.83 (0.01)	0.46	0.84 (0.01)
	500		0.05	0.27 (0.05)	0.17	0.33 (0.00)	0.12	0.79 (0.04)	0.33	0.83 (0.00)
	250		0.04	0.32 (0.02)	0.13	0.33 (0.00)	0.14	0.33 (0.00)	0.15	0.82 (0.01)
	100		0.04	0.37 (0.08)	0.11	0.33 (0.00)	0.12	0.33 (0.01)	0.00	0.33 (0.00)
	50		0.04	0.32 (0.02)	0.10	0.33 (0.00)	0.10	0.33 (0.00)	0.00	0.33 (0.00)
	10		0.04	0.43 (0.16)	0.09	0.38 (0.09)	0.09	0.33 (0.00)	0.00	0.33 (0.00)

Table 6.5: BLEU scores for the training partition of each dataset and downstream macro-averaged test F_1 performance, with each of the four models using the Analytic Gaussian mechanism and the original test F_1 results provided for comparison. Test F_1 scores shown as "mean (standard deviation)", averaging over results using three random seeds. 'N/A' refers to configurations that we did not run for **DP-BART-PR+**, since there are no additional noisy training steps at $\varepsilon = \infty$. Higher BLEU corresponds to better performance of the rewriting model for intrinsic evaluation, higher test F_1 corresponds to better downstream performance using the rewritten dataset for training. Lower ε corresponds to better privacy.

6.5.3 Sample rewritten texts

Comparing rewritten texts across privacy budgets

Original	It slows the game performance a bit, but it's totally worth it!
$\varepsilon = 2500$	The performance of the game is a bit sluggish, but it's worth it
$\varepsilon = 1000$	It's that time of year again. But if you slow down your
$\varepsilon = 750$	It's that time of year again when we talk about kitty racing
$\varepsilon = 500$	We've all been talking about the game, but this is a bit of
$\varepsilon = 250$	12 years ago today morning morning morning, a 12-

Sample rewritten texts for varying privacy budgets, using **DP-BART-PR**+ for the Amazon dataset.

Original	i want to hear something eclectic
$\varepsilon = 2500$	The following is a list of interesting things to hear from the eclectic, eclectic, and
$\varepsilon = 1000$	i want to hear something different from what everyone else has been hearing about this week.
$\varepsilon = 750$	i want to hear something different about this mod. It's simple, but
$\varepsilon = 500$	i want to hear something like this. If you want to listen to music
$\varepsilon = 250$	In the last three year in the last time it seems to have an area of the

Sample rewritten texts for varying privacy budgets, using **DP-BART-PR**+ for the Snips dataset.

We provide sample rewritten texts from the **DP-BART-PR**+ model, comparing the difference in output across ε values on the Snips and Amazon datasets. We can see that, for different values of ε, parts of the original input sequence reappear in the rewritten output to varying degrees. For example, the first five tokens of the original Snips sample reappear in the rewritten texts at $\varepsilon = 500, 750, 1000$. At the lower ε value of 250, while the output is still in part coherent, it is no longer recognizable from the original. At the lowest ε values, there is so much noise added to the model that the output primarily consists of 'start of sequence' and 'end of sequence' tokens, resulting in an overall empty output. For the Amazon example, most rewritten tokens are different from the input, with some resemblance at $\varepsilon = 500$, but a more coherent and related output primarily at the larger $\varepsilon = 2500$.

Interestingly for these examples, while the rewritten documents are very altered from the original documents throughout, it is enough in the case of **DP-BART-PR**+ to achieve a relatively good downstream performance, such as an F_1 score of 0.65 for Snips at $\varepsilon = 500$ and 0.82 for Amazon at $\varepsilon = 250$. This is more of what we would expect from a text rewriting system, since if the original text is clearly noticeable in the rewritten output, we would strongly suspect a privacy leak.

Comparing rewritten texts across models

Original	The product doesn't work at all.
ADePT	has ! low phone unauthorised and 1 awesome 5th whatsoever pickle my canna kindle just flowed phones signup
DP-BART-CLV	""". @...???)!).. W @. W???)
DP-BART-PR	Technical precisely anticipate work-touch to enhance Resources Resources ARE/and and Science Matters/
DP-BART-PR+	"The product doesn't work at all." That is the sentiment of

Sample rewritten texts for each model type, at $\varepsilon = 750$ for the Amazon dataset.

We additionally provide sample rewritten texts from each model, at the same ε value and on the same dataset (Amazon at $\varepsilon = 750$). Here we can see that the **DP-BART-PR+** model output is the most similar to the original document, being rewritten verbatim, followed by some additional output. The output sequence for **DP-BART-PR** is less coherent, but still with recognizable sequences for some token pairs, while **DP-BART-CLV** and **ADePT** have output that is seemingly random.

6.6 Discussion

6.6.1 Reducing noise for text rewriting with LDP

We have shown that it is possible to reduce the amount of noise in the LDP setting of privatized rewriting, in order to obtain more useful rewritten texts for downstream tasks. To compare **DP-BART-CLV** vs. **DP-BART-PR**, we can examine the resulting ℓ_2 sensitivity from Eqn. 6.7 ($\Delta_2 f = 2C\sqrt{n}$). Setting sequence length $l = 20$ and $C = 0.1$, as in our experiments, without pruning we have a dimensionality of $n = 768 \cdot 20 = 15360$, hence $\Delta_2 f = 2 \cdot 0.1 \cdot \sqrt{15360} \approx 24.79$. With pruning we are able to remove 76.30% of those n neurons, with only $n = 182 \cdot 20 = 3640$ remaining. The ℓ_2 sensitivity thus becomes $\Delta_2 f = 2 \cdot 0.1 \cdot \sqrt{3640} \approx 12.07$.

Plugging this into the Analytic Gaussian mechanism's noise scale calculation from Balle and Wang (2018), with $\delta = 10^{-5}$ and $\varepsilon = 500$, we have $\sigma^2 = 0.8958$ without pruning and $\sigma^2 = 0.4362$. We can therefore see that, **with DP-BART-PR, we are able to reduce the noise scale by more than half**.

6.6.2 Pre-training and computational resources

Ultimately, a very effective way to prepare a model for privatized text rewriting would be to pre-train it from scratch, being fully in control of hyperparameters such as the dimensionality n of the encoder output vectors z, which determines the ℓ_1 and ℓ_2 sensitivities from Eqns. 6.5 and 6.7, respectively. In addition, the whole model could be pre-trained with added noise and clipping mechanisms, potentially being even more robust than our approach in **DP-BART-PR+**, where we incorporate further noisy training. We noticed for **DP-BART-PR+** that the lower the ε value

we use, the more additional training iterations the model needs to properly reduce the validation loss.

This demonstrates that, also in the setting of pre-training from scratch, we would need to train for more iterations in order to reach lower ε values. This can pose serious challenges, however, for reasons of computational demand discussed in Section 6.3.2. **DP-BART-PR+** can therefore be seen as a sweet spot approach, where we only need a few additional training iterations and can still achieve a significant dimensionality reduction through pruning, as well as additional robustness to noise.

6.6.3 Domain of public training texts

In preparing the DP-BART models, it is important to take into account the domain of the public data that is used to (1) pre-train the original BART model, and (2) perform additional training iterations (**DP-BART-PR** and **DP-BART-PR+**). This will ultimately have an impact on the model's effectiveness for text privatization, depending on the nature of the downstream texts. For example, if this training data is restricted to news articles, then there may be limited performance for rewriting texts that are further from this domain, such as internet comments. Another obvious limitation is the language of the public data. If the model is trained on a monolingual English corpus, then it would not be possible to use it for rewriting texts from other languages.

The public data used for our experiments consists of news, web text, stories and books (Lewis et al., 2020a; Gokaslan and Cohen, 2019). We expect that expanding this to include more data and more varied domains will lead to better performance in a greater diversity of texts and downstream tasks.

6.6.4 What is being privatized

As we discuss throughout this thesis (e.g. Section 1.2.2 of Chapter 1, Section 4.2.1 of Chapter 4, Section 7.2 of Chapter 7), it is very important to be clear on exactly what information is being privatized when performing text rewriting with LDP. Since we are working with DP at the document level, the entire document is a 'data point', hence any choice and combination of words for a given sequence would be a unique identifier. We thus avoid the problem of having to choose what specific tokens are 'private' within the document. This is crucial, since stylistic aspects of an author can be very abstract, with subtle syntactic and vocabulary choices.

Another significant benefit of such an approach, is that we are not limiting ourselves to any specific downstream analysis (e.g. sentiment of a document), being *task agnostic*. However, this also means that, for any given document, *any other document is neighboring*, since we are in the LDP setting, as mentioned in Section 6.1. This leads us to a serious discussion on the limitations of such an approach below in Section 6.7.

An additional question arises of whether one dataset may have multiple documents associated with one individual. There are several ways to go about dealing

with this. One standard approach in differential privacy is to linearly scale the ε parameter. Thus, if there are k documents associated with a given individual, then a privacy budget of $k\varepsilon$ is accounted in total (Dwork and Roth, 2013). Another option would be to simply append all texts associated with one individual into a single 'document', rewriting this using just a single ε privacy budget.

6.7 Limitations of LDP for text rewriting

For every output document, any two inputs, no matter how similar or distinct, are considered neighboring. If we have a small sequence length of 20 tokens, with a relatively small vocabulary of 1000 words, then the total number of possible combinations if 1000^{20}, which is 10^{60}! While we compress these documents into a latent vector with a limited range and dimensionality, the strict adjacency constraints are still present. We can therefore expect an inevitable utility drop when using more reasonable ε values (e.g. $\varepsilon = 1$).

With more sophisticated architectures, we have shown that it is possible to push this ε value down to some extent. However, our lowest ε is still too high to carry over into real-world applications of privacy preservation. As outlined by Hsu et al. (2014), values of ε for different applications in the DP literature can range from 0.01 to 10. Choosing the right ε value depends on the specific queries that are computed and the nature of the data (Lee and Clifton, 2011).

For our case, the value of ε can be interpreted in the following manner. The ε-LDP mechanism that we are applying to our data makes any two input texts rewritten to be indistinguishable up to a factor of e^{ε}. More formally, *for any two input texts* x and y to our LDP model $\mathcal{M}$:

$$\frac{\Pr[\mathcal{M}(x) = z]}{\Pr[\mathcal{M}(y) = z]} \leq e^{\varepsilon}, \tag{6.11}$$

where z is a given output text rewritten by the model (see also Eqns. 2.2 and 2.3, as well as the overall discussion on the privacy loss random variable in Section 2.2.1 of Chapter 2).

This means that, when we set $\varepsilon = 250$, then any two texts will remain indistinguishable up to a factor of e^{250}. This is a very weak bound and, while it could provide some empirical privacy guarantees, on a theoretical level the privacy protection is not very strong. We can also see how this bound becomes exponentially stronger, as we decrease ε.

It may therefore make sense to take a slightly less strict approach to text adjacency, for instance moving into *domain specific* text rewriting. For example, text rewriting could be carried out for a specific dataset, with the notion of adjacency restricted to any two individuals within that dataset, hence requiring much less perturbation. The strength of the privacy guarantee, in this case, would then be very dependent on the size of the dataset (Mehner et al., 2021).

6.8 Chapter Summary

We have thus concluded our investigation into privatizing textual data in the LDP setting. We have proposed DP-BART, a set of methodologies consisting of applying various mechanisms to a pre-trained BART model, with an added differential privacy component at the end of its final encoder. The resulting system rewrites textual data, such that the rewritten output satisfies a specified LDP guarantee. The goal of these methods is to reduce the added noise to the pre-trained model in the DP setting by lowering the sensitivity of its DP module, i.e. the final encoder function, as well as being computationally feasible to implement.

We have demonstrated our method's privacy/utility trade-off, the relations between the privacy budget and dataset size, and discussed limitations of the privatized text rewriting approach as a whole, largely stemming from the *strict adjacency constraint*, in which any two possible documents are neighboring and must therefore be indistinguishable, up to a given ε privacy budget. Future research directions in this topic include utilizing large-scale pre-training to potentially reach a better privacy/utility trade-off, as well as investigating *domain specific* text rewriting for relaxing these strict requirements of the LDP approach.

Chapter 7

Conclusion

We have investigated a relatively new subfield of natural language processing, namely privacy-preserving NLP. Specifically, we have worked in the framework of differential privacy, a very promising direction that has recently been taking off in NLP and the broader field of machine learning in general.

7.1 Summary

In our investigation of differentially private NLP, we have looked into three main research questions, dividing into two primary subtopics. The first of these is the *privatization of NLP models*, protecting the information of individuals in the datasets used to train them. For this subtopic, we first investigate the privatization of text classification models, with our initial research question (RQ1) addressing how to privatize text classification models that operate on graph datasets. Using the standard method of applying DP to machine learning models, DP-SGD, we saw that new challenges emerge in the graph setting, notably the issues of dealing with large graphs that in turn lead to a large amount of added DP noise. We demonstrate our solution to this problem by means of our random graph splitting technique, avoiding queries on the graph, with which we achieve a good privacy/utility trade-off.

Second, we move back into the 'standard' NLP setting, looking into privatization strategies for a variety of common NLP models and tasks. Despite the ubiquity of the DP-SGD methodology, the NLP community does not have a thorough understanding of how it fares in these various settings. In performing this investigation, we find that there is no clear strategy that can be used for applying differential privacy for different models and tasks, with each configuration requiring its own specific strategies for achieving the best privacy/utility trade-off. However, we do note certain patterns that are important to keep in mind, including the negative impacts of skewed class distributions on model performance with DP-SGD, as well as the benefits of distilled variants of larger language models in the DP setting with regards to performance and computational efficiency.

The second subtopic that we delve into is differentially private text rewriting. In contrast to the above research topics that work in the global DP setting, here we are in the local DP setting, which is a stronger form of DP that does not require a trusted aggregator, but instead allows each individual to apply the DP mechanism to their own data point. Here we first tackle the problem of reproducibility and transparency that is present in this subfield. We develop the `DP-Rewrite` framework, which is an open-source, modular, and customizable framework for DP text rewriting. Our goal is to provide the community with a platform to carry out research in a transparent and reproducible manner that is in line with tackling the points we stress below on various pitfalls that can be encountered when performing DP research.

Finally, we design the DP-BART series of models, which improve on previous baselines in terms of the privacy/utility trade-off, aiming to reduce the large amount of perturbation that is required in the LDP setting, especially for transformer-based models. In this investigation, we discuss in detail the major problems and limitations of the LDP framework for textual data, largely stemming from the strict text adjacency constraint, and considering possible remedies for this.

7.2 Pitfalls to avoid when applying DP to the NLP domain

Despite the increased adoption of differential privacy, there are several notable pitfalls that can occur which the NLP community needs to be aware of:

1. DP guarantees may not be satisfied by a proposed mechanism.

2. Interpretation of ε is not always clear.

3. Unit of privatization is not always clear.

Regarding (1), due to the formal nature of DP as providing a formal, information-theoretic privacy guarantee on some data, it is certainly possible for a well-designed DP mechanism to have an unexpected miscalculation in its theoretical properties that breaks this intended privacy guarantee. While empirical verifications such as various attacks on the DP model may shed light on whether a given DP mechanism violates its claimed DP guarantees, they cannot prove whether it satisfies those guarantees. This makes it a difficult problem to verify the correctness of a proposed mechanism, demonstrating the vital importance of transparency and reproducibility, in order for the wider community to test and verify the privatized model.

For (2), as a community it is very important to be clear on what ε means, both in the intuitive and mathematical sense. Although it roughly translates to the "level of privacy guarantee", there are several important properties that are crucial to keep in mind: (a) A linear increase in ε is an exponential decrease in the privacy guarantee, (b) high ε values do not provide a strong privacy guarantee on a theoretical level, even if they may seem to do so on an empirical level, (c) all low values of ε are roughly the same (e.g. $\varepsilon < 0.5$), and (d) a 'good' value of ε is very difficult to

determine, with only a few studies investigating this point. For point (d), we can say that a fairly strong privacy guarantee can be achieved at $\varepsilon = \ln 3$, based on the randomized response technique using fair coin flips.

It is additionally important to note for (2) that sometimes the proposed DP mechanism of a given study does not always provide clear ε values in its privacy/utility trade-off demonstration when reporting results of the study. Other times, even though ε values are clearly stated, they differ from their original definition in standard differential privacy (e.g. metric DP), such that it is not always clear how a given ε value can be interpreted in the original DP sense. It is therefore crucial for investigations on NLP tasks in the DP setting to be clear on the exact ε values that are used, and how they may be different from the classic DP guarantee.

Finally, for (3), as a community we need to be very clear on what exactly we are privatizing. This is strongly related to the previous point, since an ε bound will be with respect to some unit of privatization. In contrast to the simpler case of a database of individuals, with one row corresponding to one person, in the case of natural language this problem is far from simple. We therefore need further theoretical investigations into what it means for a given piece of text to be privatized. If we apply DP at the document level, as in most of this thesis, then we are treating each document as corresponding to one given individual. While this may work for developing DP mechanisms for NLP tasks, when deploying these methods, it is very important to take into account additional complications, such as multiple people associated with one document, multiple documents associated with one person, and so forth. In the former case, the guarantees of differential privacy would need to be applied at a more fine-grained level, such as for individual tokens (e.g. specific named entities, depending on the nature of the data). The latter case can utilize additional techniques from differential privacy, such as group privacy. Overall, we need to be aware of the goals of our privacy guarantee on the text, including what constitutes a single individual in our data.

As we stress throughout this thesis, one of the main remedies to avoiding the above pitfalls and making sure that we are carrying out DP research correctly is to have transparency and reproducibility in our proposed methodologies, as discussed in Chapter 5. This means having source code available, clearly stating the ε values used, the unit of privatization, as well as all details on the proposed algorithm.

7.3 Future work

We conclude by discussing possible research directions that may be worthwhile to pursue for private NLP. First, regarding model-level privacy, while DP-SGD has become one of the most common ways to apply DP to machine learning, it is certainly not the only available option for developing a differentially private system. There may be various alternatives apart from gradient-based privatization methods which may help to improve the current state of the art with respect to the privacy/utility trade-off, such as privatizing model-internal representations in the global DP setting, as well as investigating in more detail other common proposed frameworks for private

machine learning, such as PATE (Papernot et al., 2017). It would especially be interesting to look for more NLP-specific solutions to applying differential privacy for the NLP pipeline.

For text-level differential privacy, there are also several interesting directions that may further improve on the current best methodologies. This primarily relates to further reducing the sensitivity of pure LDP text rewriting. In addition to looking into more sophisticated neuron pruning methods for achieving a lower sensitivity of the DP mechanism on a model's internal representations, other noisy training methods can also be tested, such as different schedules for gradually increasing the amount of noise during model training, as a potential enhancement to the DP-BART-PR+ configuration of Chapter 6. Another possibility is to pre-train a transformer model from scratch with both clipping and noise, which despite significantly reducing the computational efficiency of the method, could potentially result in a better privacy/utility trade-off. As mentioned in Chapter 6, task- or dataset-specific text rewriting may also reduce the sensitivity of the model, relaxing the strict adjacency constraint that is present in the LDP setting. Another option for reducing DP noise due to this constraint is to apply DP at a more fine-grained level with respect to what is considered private in a document, as opposed to treating the entire document as a single private data point.

As mentioned above, for any future work on this topic, we stress the importance of being clear on what exactly is being privatized, as well as always reporting the exact ε values for which a private configuration is being reported. In addition to document-level DP guarantees, it would be immensely beneficial to find a rigorous way to perform more fine-grained privatization of textual representations (e.g. document-level, token-level, token-level with the same sensitive token appearing in multiple documents, and so forth). This would help to resolve the question of what is private information in text, as well as reduce the amount of noise that needs to be added in the DP setting due to lower sensitivity, in turn improving utility of these DP mechanisms. One significant issue to overcome for this last point is that the exact definition of what is private information may be different for each person, making fine-grained privacy analysis very difficult over a dataset of many individuals. A possible remedy for this would then be to find a formal, legal standard that can be set, which would then need to be followed when designing a DP mechanism for textual data and tasks in the NLP domain.

Appendix A

Data Handling

According to the DFG's "Principles for the Handling of Research Data",[1] we ensured that the research data and experimental software related to this dissertation is publicly available when possible and archived for long-term preservation. The following is a list of software that has been made available for the scientific community. Each repository contains details on the datasets utilized for the corresponding investigation, as well as licensing information.

- Chapter 3: https://github.com/trusthlt/privacy-preserving-gcn
- Chapter 4: https://github.com/trusthlt/dp-across-nlp-tasks
- Chapter 5: https://github.com/trusthlt/dp-rewrite
- Chapter 6: https://github.com/trusthlt/dp-bart-private-rewriting

All publications related to this dissertation are publicly available in the ACL Anthology:

- Chapter 3: https://aclanthology.org/2022.lrec-1.36/
- Chapter 4: https://aclanthology.org/2022.emnlp-main.496/
- Chapter 5: https://aclanthology.org/2022.coling-1.258/
- Chapter 6: https://aclanthology.org/2023.findings-acl.874/

Finally, all research results of the above publications are documented in the present dissertation, which is archived by the Universitäts- und Landesbibliothek Darmstadt.

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

List of Figures

List of Tables

Bibliography

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016a. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, page 265–283, USA. USENIX Association.

Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016b. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318.

Samuel Adams, David Melanson, and Martine De Cock. 2021. Private text classification with convolutional neural networks. In *Proceedings of the Third Workshop on Privacy in Natural Language Processing*, pages 53–58.

Rohan Anil, Badih Ghazi, Vineet Gupta, Ravi Kumar, and Pasin Manurangsi. 2022. Large-scale differentially private BERT. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6481–6491, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jimmy Ba and Rich Caruana. 2014. Do deep nets really need to be deep? *Advances in neural information processing systems*, 27.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Eugene Bagdasaryan, Omid Poursaeed, and Vitaly Shmatikov. 2019. Differential Privacy Has Disparate Impact on Model Accuracy. In *Advances in Neural Information Processing Systems 32*, pages 15479–15488, Vancouver, Canada. Curran Associates, Inc.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

Borja Balle and Yu-Xiang Wang. 2018. Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In *International Conference on Machine Learning*, pages 394–403. PMLR.

Raef Bassily, Adam Smith, and Abhradeep Thakurta. 2014. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th annual symposium on foundations of computer science*, pages 464–473. IEEE.

Priyam Basu, Tiasa Singha Roy, Rakshit Naidu, and Zumrut Muftuoglu. 2021. Privacy enabled financial text classification using differential privacy and federated learning. In *Proceedings of the Third Workshop on Economics and Natural Language Processing*, pages 50–55, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ghazaleh Beigi, Kai Shu, Ruocheng Guo, Suhang Wang, and Huan Liu. 2019. Privacy preserving text representation learning. In *Proceedings of the 30th ACM Conference on Hypertext and Social Media*, HT '19, page 275–276, New York, NY, USA. Association for Computing Machinery.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

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.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Haohan Bo, Steven H. H. Ding, Benjamin C. M. Fung, and Farkhund Iqbal. 2021. ER-AE: Differentially private text generation for authorship anonymization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3997–4007, Online. Association for Computational Linguistics.

Rishi Bommasani, Steven Wu, and Xanda Schofield. 2019. Towards private synthetic text generation. In *NeurIPS 2019 Machine Learning with Guarantees Workshop*.

Léon Bottou, Corinna Cortes, John S Denker, Harris Drucker, Isabelle Guyon, Larry D Jackel, Yann LeCun, Urs A Muller, Edward Sackinger, Patrice Simard, et al. 1994. Comparison of classifier methods: a case study in handwritten digit recognition. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 3-Conference C: Signal Processing (Cat. No. 94CH3440-5)*, volume 2, pages 77–82. IEEE.

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.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, et al. 2018. Jax: composable transformations of python+ numpy programs.

Louis Brandeis and Samuel Warren. 1890. The right to privacy. *Harvard law review*, 4(5):193–220.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Thang Nguyen Bui and Curt Jones. 1992. Finding good approximate vertex and edge partitions is np-hard. *Information Processing Letters*, 42(3):153–159.

Mark Bun and Thomas Steinke. 2016. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography: 14th International Conference, TCC 2016-B, Beijing, China, October 31-November 3, 2016, Proceedings, Part I*, pages 635–658. Springer.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluating the role of bleu in machine translation research. In *11th conference of the european chapter of the association for computational linguistics*, pages 249–256.

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. 2019. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *USENIX Security Symposium*, volume 267.

Ricardo Silva Carvalho, Theodore Vasiloudis, and Oluwaseyi Feyisetan. 2021. Brr: Preserving privacy of text data efficiently on device. *arXiv preprint arXiv:2107.07923*.

Ricardo Silva Carvalho, Theodore Vasiloudis, Oluwaseyi Feyisetan, and Ke Wang. 2023. Tem: high utility metric differential privacy on text. In *Proceedings of the 2023 SIAM International Conference on Data Mining (SDM)*, pages 883–890. SIAM.

Augustin Cauchy et al. 1847. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538.

Konstantinos Chatzikokolakis, Miguel E Andrés, Nicolás Emilio Bordenabe, and Catuscia Palamidessi. 2013. Broadening the scope of differential privacy using metrics. In *Privacy Enhancing Technologies: 13th International Symposium, PETS 2013, Bloomington, IN, USA, July 10-12, 2013. Proceedings 13*, pages 82–102. Springer.

Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. 2011. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(3).

Tianyu Chen, Hangbo Bao, Shaohan Huang, Li Dong, Binxing Jiao, Daxin Jiang, Haoyi Zhou, Jianxin Li, and Furu Wei. 2022. The-x: Privacy-preserving trans-

former inference with homomorphic encryption. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3510–3520.

Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 2019. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 257–266.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Maximin Coavoux, Shashi Narayan, and Shay B. Cohen. 2018. Privacy-preserving Neural Representations of Text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1–10, Brussels, Belgium. Association for Computational Linguistics.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. 2015. Binaryconnect: Training deep neural networks with binary weights during propagations. *Advances in neural information processing systems*, 28.

Ronald Cramer, Ivan Bjerre Damgrd, and Jesper Buus Nielsen. 2015. Secure multiparty computation and secret sharing.

Deborah A Dahl, Madeleine Bates, Michael K Brown, William M Fisher, Kate Hunicke-Smith, David S Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the atis task: The atis-3 corpus. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

Soham De, Leonard Berrada, Jamie Hayes, Samuel L Smith, and Borja Balle. 2022. Unlocking high-accuracy differentially private image classification through scale. *arXiv preprint arXiv:2204.13650*.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2019. Question Answering by Rea-

soning Across Documents with Graph Convolutional Networks. 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 2306–2317, Minneapolis, Minnesota. Association for Computational Linguistics.

Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. 2014. Exploiting linear structure within convolutional networks for efficient evaluation. *Advances in neural information processing systems*, 27.

Damien Desfontaines and Balázs Pejó. 2020. SoK: Differential privacies. *Proceedings on Privacy Enhancing Technologies*, 2020(2):288–313.

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.

Cynthia Dwork. 2011. The promise of differential privacy: a tutorial on algorithmic techniques. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science, D (Oct. 2011)*, pages 1–2. Citeseer.

Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006a. Our data, ourselves: Privacy via distributed noise generation. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 486–503. Springer.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006b. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer.

Cynthia Dwork and Aaron Roth. 2013. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3-4):211–407.

Cynthia Dwork and Guy N Rothblum. 2016. Concentrated differential privacy. *arXiv preprint arXiv:1603.01887*.

Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. 2010. Boosting and differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 51–60. IEEE.

Yanai Elazar and Yoav Goldberg. 2018. Adversarial removal of demographic attributes from text data. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 11–21.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Maël Fabien, Esaú Villatoro-Tello, Petr Motlicek, and Shantipriya Parida. 2020. Bertaa: Bert fine-tuning for authorship attribution. In *Proceedings of the 17th International Conference on Natural Language Processing (ICON)*, pages 127–137.

Tom Farrand, Fatemehsadat Mireshghallah, Sahib Singh, and Andrew Trask. 2020. Neither Private Nor Fair: Impact of Data Imbalance on Utility and Fairness in Differential Privacy. In *Proceedings of the 2020 Workshop on Privacy-Preserving Machine Learning in Practice*, pages 15–19, Virtual conference. ACM.

Oluwaseyi Feyisetan, Abhinav Aggarwal, Zekun Xu, and Nathanael Teissier. 2021. Research challenges in designing differentially private text generation mechanisms. In *The International FLAIRS Conference Proceedings*, volume 34.

Oluwaseyi Feyisetan, Borja Balle, Thomas Drake, and Tom Diethe. 2020. Privacy- and utility-preserving textual analysis via calibrated multivariate perturbations. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 178–186.

Oluwaseyi Feyisetan, Tom Diethe, and Thomas Drake. 2019. Leveraging hierarchical representations for preserving privacy and utility in text. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 210–219. IEEE.

Oluwaseyi Feyisetan and Shiva Kasiviswanathan. 2021. Private release of text embedding vectors. In *Proceedings of the First Workshop on Trustworthy Natural Language Processing*, pages 15–27, Online. Association for Computational Linguistics.

J.R. Firth. 1957. *A Synopsis of Linguistic Theory, 1930-1955.*

Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635.*

Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1322–1333.

Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. 2014. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pages 17–32.

Charles Fried. 1990. Privacy: A rational context. In *Computers, ethics, & society*, pages 51–63.

Max Friedrich, Arne Köhn, Gregor Wiedemann, and Chris Biemann. 2019. Adversarial learning of privacy-preserving text representations for de-identification of medical records. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5829–5839, Florence, Italy. Association for Computational Linguistics.

Roy Frostig, Matthew James Johnson, and Chris Leary. 2018. Compiling machine learning programs via high-level tracing. *Systems for Machine Learning*, 4(9).

Ruth Gavison. 1980. Privacy and the limits of law. *The Yale law journal*, 89(3):421–471.

Suyu Ge, Fangzhao Wu, Chuhan Wu, Tao Qi, Yongfeng Huang, and Xing Xie. 2020. Fedner: Privacy-preserving medical named entity recognition with federated learning. *arXiv preprint arXiv:2003.09288*.

Craig Gentry. 2009. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178.

C. Lee Giles, Kurt D. Bollacker, and Steve Lawrence. 1998. CiteSeer: An Automatic Citation Indexing System. In *Proceedings of the third ACM conference on Digital Libraries*, pages 89–98, Pittsburgh, PA, USA. ACM Press.

Aaron Gokaslan and Vanya Cohen. 2019. Openwebtext corpus. `http://Skylion007.github.io/OpenWebTextCorpus`.

Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. 2018. Slot-gated modeling for joint slot filling and intent prediction. 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 753–757.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Google. XLA: Optimizing Compiler for Machine Learning. `https://www.tensorflow.org/xla`. License: Apache 2.0 License.

Google. 2018. Tensorflow privacy.

Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129:1789–1819.

Felix Gräßer, Surya Kallumadi, Hagen Malberg, and Sebastian Zaunseder. 2018. Aspect-based sentiment analysis of drug reviews applying cross-domain and cross-data learning. In *Proceedings of the 2018 International Conference on Digital Health*, pages 121–125.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Bernard G Greenberg, Abdel-Latif A Abul-Ela, Walt R Simmons, and Daniel G Horvitz. 1969. The unrelated question randomized response model: Theoretical framework. *Journal of the American Statistical Association*, 64(326):520–539.

Saurabh Gupta, Judy Hoffman, and Jitendra Malik. 2016. Cross modal distillation for supervision transfer. In *29th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*, pages 2827–2836. IEEE Computer Society.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation Artifacts in Natural Language Inference Data. 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 107–112, New Orleans, LA. Association for Computational Linguistics.

Ivan Habernal. 2021. When differential privacy meets NLP: The devil is in the detail. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1522–1528, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ivan Habernal. 2022. How reparametrization trick broke differentially-private text representation learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 771–777, Dublin, Ireland. Association for Computational Linguistics.

Will Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems 30*, pages 1024–1034, Long Beach, CA, USA. Curran Associates, Inc.

Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28.

Babak Hassibi, David G Stork, and Gregory J Wolff. 1993. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, pages 293–299. IEEE.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th international conference on world wide web*, pages 507–517.

Steven Hill, Zhimin Zhou, Lawrence K Saul, and Hovav Shacham. 2016. On the (In) effectiveness of Mosaicing and Blurring as Tools for Document Redaction. *Proc. Priv. Enhancing Technol.*, 2016(4):403–417.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. 2017. Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, page 603–618, Dallas, TX, USA. Association for Computing Machinery.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Nils Homer, Szabolcs Szelinger, Margot Redman, David Duggan, Waibhav Tembe, Jill Muehling, John V Pearson, Dietrich A Stephan, Stanley F Nelson, and David W Craig. 2008. Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays. *PLoS genetics*, 4(8):e1000167.

Shlomo Hoory, Amir Feder, Avichai Tendler, Sofia Erell, Alon Peled-Cohen, Itay Laish, Hootan Nakhost, Uri Stemmer, Ayelet Benjamini, Avinatan Hassidim, and Yossi Matias. 2021. Learning and Evaluating a Differentially Private Pre-trained Language Model. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1178–1189, Punta Cana, Dominican Republic. Association for Computational Linguistics.

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:1704.04861*.

Justin Hsu, Marco Gaboardi, Andreas Haeberlen, Sanjeev Khanna, Arjun Narayan, Benjamin C Pierce, and Aaron Roth. 2014. Differential privacy: An economic method for choosing epsilon. In *2014 IEEE 27th Computer Security Foundations Symposium*, pages 398–410. IEEE.

Tao Huang and Hong Chen. 2021. Improving privacy guarantee and efficiency of Latent Dirichlet Allocation model training under differential privacy. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 143–152, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Timour Igamberdiev, Thomas Arnold, and Ivan Habernal. 2022. DP-rewrite: Towards reproducibility and transparency in differentially private text rewriting. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 2927–2933, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Timour Igamberdiev and Ivan Habernal. 2022. Privacy-preserving graph convolutional networks for text classification. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 338–350.

Timour Igamberdiev and Ivan Habernal. 2023. Dp-bart for privatized text rewriting under local differential privacy. In *Findings of the Association for Computational Linguistics: ACL 2023"*. Association for Computational Linguistics.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr.

Prateek Jain, Pravesh Kothari, and Abhradeep Thakurta. 2012. Differentially private online learning. In *Conference on Learning Theory*, pages 24–1. JMLR Workshop and Conference Proceedings.

Abhik Jana and Chris Biemann. 2021. An Investigation towards Differentially Private Sequence Tagging in a Federated Framework. In *Proceedings of the Third*

Workshop on Privacy in Natural Language Processing, pages 30–35, 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.

Armand Joulin, Édouard Grave, Piotr Bojanowski, and Tomáš Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431.

George Karypis and Vipin Kumar. 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392.

Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2013. Analyzing graphs with node differential privacy. In *Theory of Cryptography Conference*, pages 457–476. Springer.

Gavin Kerrigan, Dylan Slack, and Jens Tuyls. 2020. Differentially Private Language Models Benefit from Public Pre-training. In *Proceedings of the Second Workshop on Privacy in NLP*, pages 39–45, Online. Association for Computational Linguistics.

Daniel Kifer, Adam Smith, and Abhradeep Thakurta. 2012. Private convex empirical risk minimization and high-dimensional regression. In *Conference on Learning Theory*, pages 25–1. JMLR Workshop and Conference Proceedings.

Donggyu Kim, Garam Lee, and Sungwoo Oh. 2022. Toward privacy-preserving text embedding similarity with homomorphic encryption. In *Proceedings of the Fourth Workshop on Financial Technology and Natural Language Processing (FinNLP)*, pages 25–36.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. *2nd International Conference on Learning Representations, ICLR*.

Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of International Conference on Learning Representations (ICLR 2017)*, pages 1–14, Toulon, France.

Oleksandra Klymenko, Stephen Meisenbacher, and Florian Matthes. 2022. Differential privacy in natural language processing the story so far. In *Proceedings of the Fourth Workshop on Privacy in Natural Language Processing*, pages 1–11, Seattle, United States. Association for Computational Linguistics.

Satyapriya Krishna, Rahul Gupta, and Christophe Dupuy. 2021. ADePT: Auto-encoder based differentially private text transformation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational*

Linguistics: Main Volume, pages 2435–2439, Online. Association for Computational Linguistics.

John K Kruschke and Javier R Movellan. 1991. Benefits of gain: Speeded learning and minimal hidden layers in back-propagation networks. *IEEE Transactions on systems, Man, and Cybernetics*, 21(1):273–280.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Yann LeCun, John Denker, and Sara Solla. 1989. Optimal brain damage. *Advances in neural information processing systems*, 2.

Garam Lee, Minsoo Kim, Jai Hyun Park, Seung-won Hwang, and Jung Hee Cheon. 2022. Privacy-preserving text classification on BERT embeddings with homomorphic encryption. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3169–3175, Seattle, United States. Association for Computational Linguistics.

Jaewoo Lee and Chris Clifton. 2011. How much is enough? choosing ε for differential privacy. In *International Conference on Information Security*, pages 325–340. Springer.

Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`.

Mike Lewis and Angela Fan. 2019. Generative question answering: Learning to answer the whole question. In *International Conference on Learning Representations*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2020a. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020b. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and William B Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

Xuechen Li, Florian Tramer, Percy Liang, and Tatsunori Hashimoto. 2022. Large language models can be strong differentially private learners. In *International Conference on Learning Representations*.

Yitong Li, Timothy Baldwin, and Trevor Cohn. 2018. Towards robust and privacy-preserving text representations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 25–30.

Pierre Lison, Ildikó Pilán, David Sanchez, Montserrat Batet, and Lilja Øvrelid. 2021. Anonymisation models for text data: State of the art, challenges and future directions. 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 4188–4203, Online. Association for Computational Linguistics.

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*.

Hans Peter Luhn. 1957. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of research and development*, 1(4):309–317.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Lingjuan Lyu, Xuanli He, and Yitong Li. 2020a. Differentially private representation for NLP: Formal guarantee and an empirical study on privacy and fairness. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2355–2365, Online. Association for Computational Linguistics.

Lingjuan Lyu, Yitong Li, Xuanli He, and Tong Xiao. 2020b. Towards Differentially Private Text Representations. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1813–1816, Virtual conference. ACM.

Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150.

Avinash Madasu and Vijjini Anvesh Rao. 2019. Sequential Learning of Convolutional Features for Effective Text Classification. 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 5657–5666, Hong Kong, China. Association for Computational Linguistics.

Gaurav Maheshwari, Pascal Denis, Mikaela Keller, and Aurélien Bellet. 2022. Fair NLP models with differentially private text encoders. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6913–6930, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Diego Marcheggiani, Jasmijn Bastings, and Ivan Titov. 2018. Exploiting semantics in neural machine translation with graph convolutional networks. 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 486–492, New Orleans, Louisiana. Association for Computational Linguistics.

Justus Mattern, Zhijing Jin, Benjamin Weggenmann, Bernhard Schoelkopf, and Mrinmaya Sachan. 2022a. Differentially private language models for secure data sharing. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4860–4873, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Justus Mattern, Benjamin Weggenmann, and Florian Kerschbaum. 2022b. The Limits of Word Level Differential Privacy. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 867–881, Seattle, WA, USA. Association for Computational Linguistics.

Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163.

Warren S McCulloch and Walter Pitts. 1943. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR.

H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2018. Learning Differentially Private Recurrent Language Models. In *Proceedings of the 6th International Conference on Learning Representations*, pages 1–14, Vancouver, BC, Canada.

Frank McSherry and Kunal Talwar. 2007. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 94–103. IEEE.

Casey Meehan, Khalil Mrini, and Kamalika Chaudhuri. 2022. Sentence-level privacy for document embeddings. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3367–3380, Dublin, Ireland. Association for Computational Linguistics.

Luise Mehner, Saskia Nuñez von Voigt, and Florian Tschorsch. 2021. Towards explaining epsilon: A worst-case study of differential privacy risks. In *2021 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 328–331. IEEE.

Kentaro Mihara, Ryohei Yamaguchi, Miguel Mitsuishi, and Yusuke Maruyama. 2020. Neural network training with homomorphic encryption. *arXiv preprint arXiv:2012.13552*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Fatemehsadat Mireshghallah, Huseyin Inan, Marcello Hasegawa, Victor Rühle, Taylor Berg-Kirkpatrick, and Robert Sim. 2021. Privacy regularization: Joint privacy-utility optimization in LanguageModels. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3799–3807, Online. Association for Computational Linguistics.

Ilya Mironov. 2017. Rényi differential privacy. In *2017 IEEE 30th computer security foundations symposium (CSF)*, pages 263–275. IEEE.

Adam D Moore. 2008. Defining privacy. *Journal of Social Philosophy*, 39(3):411–428.

Michael C Mozer. 1995. A focused backpropagation algorithm for temporal pattern recognition. *Backpropagation: Theory, architectures, and applications*, 137.

Subhabrata Mukherjee, Ahmed Hassan Awadallah, and Jianfeng Gao. 2021. Xtremedistiltransformers: Task transfer for task-agnostic distillation.

Subhabrata Mukherjee and Ahmed Hassan Awadallah. 2020. XtremeDistil: Multi-stage Distillation for Massive Multilingual Models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2221–2234, Online. Association for Computational Linguistics.

Arvind Narayanan and Vitaly Shmatikov. 2008. Robust de-anonymization of large sparse datasets. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 111–125. IEEE.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajic, Christopher D Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4034–4043.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual Name Tagging and Linking for 282 Languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1946–1958, Vancouver, Canada. Association for Computational Linguistics.

Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. 2017. Semi-supervised knowledge transfer for deep learning from private training data. *Proceedings of the 5th International Conference on Learning Representations (ICLR)*.

Nicolas Papernot, Abhradeep Thakurta, Shuang Song, Steve Chien, and Úlfar Erlingsson. 2021. Tempered sigmoid activations for deep learning with differential privacy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9312–9321.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the*

40th annual meeting of the Association for Computational Linguistics, pages 311–318.

William A Parent. 1983. Privacy, morality, and the law. *Philosophy & Public Affairs*, pages 269–288.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. PMLR.

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.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Bryan Perozzi and Steven Skiena. 2015. Exact Age Prediction in Social Networks. In *Proceedings of the 24th International Conference on World Wide Web - WWW '15 Companion*, pages 91–92, Florence, Italy. ACM Press.

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.

Victor Petren Bach Hansen, Atula Tejaswi Neerkaje, Ramit Sawhney, Lucie Flek, and Anders Sogaard. 2022. The impact of differential privacy on group disparity mitigation. In *Proceedings of the Fourth Workshop on Privacy in Natural Language Processing*, pages 12–12, Seattle, United States. Association for Computational Linguistics.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2089–2096.

Richard Plant, Dimitra Gkatzia, and Valerio Giuffrida. 2021. CAPE: Context-aware private embeddings for private language learning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7970–7978, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Robert Podschwadt and Daniel Takabi. 2020. Classification of encrypted word embeddings using recurrent neural networks. In *PrivateNLP@ WSDM*, pages 27–31.

Antonio Polino, Razvan Pascanu, and Dan Alistarh. 2018. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*.

Natalia Ponomareva, Jasmijn Bastings, and Sergei Vassilvitskii. 2022. Training text-to-text transformers with privacy guarantees. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2182–2193, Dublin, Ireland. Association for Computational Linguistics.

Tao Qi, Fangzhao Wu, Chuhan Wu, Yongfeng Huang, and Xing Xie. 2020. Privacy-preserving news recommendation model learning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1423–1432.

Tao Qi, Fangzhao Wu, Chuhan Wu, Yongfeng Huang, and Xing Xie. 2021. Unifedrec: A unified privacy-preserving news recommendation framework for model training and online serving. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1438–1448.

Chen Qu, Weize Kong, Liu Yang, Mingyang Zhang, Michael Bendersky, and Marc Najork. 2021. Natural language understanding with privacy-preserving bert. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 1488–1497.

James Rachels. 1975. Why privacy is important. *Philosophy & Public Affairs*, pages 323–333.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

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):9.

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. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Afshin Rahimi, Trevor Cohn, and Timothy Baldwin. 2018. Semi-supervised User Geolocation via Graph Convolutional Networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2009–2019, Melbourne, Australia. Association for Computational Linguistics.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know What You Don't Know: Unanswerable Questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In *Third Workshop on Very Large Corpora*.

Adwait Ratnaparkhi. 1998. Maximum entropy models for natural language ambiguity resolution.

Devin Reich, Ariel Todoki, Rafael Dowsley, Martine De Cock, et al. 2019. Privacy-preserving classification of personal text messages with secure multi-party computation. *Advances in Neural Information Processing Systems*, 32.

Nils Reimers 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 3980–3990, Hong Kong, China. Association for Computational Linguistics.

Amanda Resende, Davis Railsback, Rafael Dowsley, Anderson C. A. Nascimento, and Diego F. Aranha. 2021. Fast privacy-preserving text classification based on secure multiparty computation. Cryptology ePrint Archive, Paper 2021/069. https://eprint.iacr.org/2021/069.

Herbert Robbins and Sutton Monro. 1951. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A Primer in BERTology: What We Know About How BERT Works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2015. Fitnets: Hints for thin deep nets. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Frank Rosenblatt. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Benjamin IP Rubinstein, Peter L Bartlett, Ling Huang, and Nina Taft. 2012. Learning in a large function space: Privacy-preserving mechanisms for svm learning. *Journal of Privacy and Confidentiality*, 4(1):65–100.

Sebastian Ruder. 2016. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1985. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Devendra Singh Sachan, Yuhao Zhang, Peng Qi, and William Hamilton. 2020. Do

syntax trees help pre-trained transformers extract information? *arXiv preprint arXiv:2008.09084*.

Sina Sajadmanesh and Daniel Gatica-Perez. 2020. When Differential Privacy Meets Graph Neural Networks. *arXiv preprint*.

Pierangela Samarati and Latanya Sweeney. 1998. Generalizing data to provide anonymity when disclosing information. In *PODS*, volume 98, pages 10–1145.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective Classification in Network Data. *AI Magazine*, 29(3):93.

Manuel Senge, Timour Igamberdiev, and Ivan Habernal. 2022. One size does not fit all: Investigating strategies for differentially-private learning across NLP tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 7340–7353, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Weiyan Shi, Aiqi Cui, Evan Li, Ruoxi Jia, and Zhou Yu. 2022a. Selective differential privacy for language modeling. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2848–2859, Seattle, United States. Association for Computational Linguistics.

Weiyan Shi, Ryan Shea, Si Chen, Chiyuan Zhang, Ruoxi Jia, and Zhou Yu. 2022b. Just fine-tune twice: Selective differential privacy for large language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6327–6340, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Akhil Shiju and Zhe He. 2022. Classifying drug ratings using user reviews with transformer-based language models. In *2022 IEEE 10th International Conference on Healthcare Informatics (ICHI)*, pages 163–169.

Reza Shokri and Vitaly Shmatikov. 2015. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1310–1321.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE.

Natalia Silveira, Timothy Dozat, Marie Catherine De Marneffe, Samuel R. Bowman, Miriam Connor, John Bauer, and Christopher D. Manning. 2014. A gold standard dependency corpus for English. In *Proceedings of the Ninth International*

Conference on Language Resources and Evaluation (LREC'14), pages 2897–2904, Reykjavik, Iceland. European Language Resources Association (ELRA).

Congzheng Song and Vitaly Shmatikov. 2019. Auditing data provenance in text-generation models. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 196–206.

Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. 2013. Stochastic gradient descent with differentially private updates. In *2013 IEEE global conference on signal and information processing*, pages 245–248. IEEE.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21.

Pranav Subramani, Nicholas Vadivelu, and Gautam Kamath. 2021. Enabling fast differentially private sgd via just-in-time compilation and vectorization. *Advances in Neural Information Processing Systems*, 34:26409–26421.

Dianbo Sui, Yubo Chen, Jun Zhao, Yantao Jia, Yuantao Xie, and Weijian Sun. 2020. Feded: Federated learning via ensemble distillation for medical relation extraction. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pages 2118–2128.

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.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Latanya Sweeney. 1997. Weaving technology and policy together to maintain confidentiality. *The Journal of Law, Medicine & Ethics*, 25(2-3):98–110.

Latanya Sweeney. 2002. k-anonymity: A model for protecting privacy. *International journal of uncertainty, fuzziness and knowledge-based systems*, 10(05):557–570.

Lubos Takac and Michal Zabovsky. 2012. Data analysis in public social networks. In *International scientific conference and international workshop present day trends of innovations*, Łomża, Poland.

Herman T Tavani. 2007. Philosophical theories of privacy: Implications for an adequate online privacy policy. *Metaphilosophy*, 38(1):1–22.

Lucien Tesnière. 1959. *Éléments de syntaxe structurale*. Klincksieck, Paris.

Om Dipakbhai Thakkar, Swaroop Ramaswamy, Rajiv Mathews, and Francoise Beaufays. 2021. Understanding unintended memorization in language models under federated learning. In *Proceedings of the Third Workshop on Privacy in Natural Language Processing*, pages 1–10.

Aleena Thomas, David Ifeoluwa Adelani, Ali Davody, Aditya Mogadala, and Dietrich Klakow. 2020. Investigating the impact of pre-trained word embeddings

on memorization in neural networks. In *Text, Speech, and Dialogue: 23rd International Conference, TSD 2020, Brno, Czech Republic, September 8–11, 2020, Proceedings 23*, pages 273–281. Springer.

Paul B Thompson. 2001. Privacy, secrecy and security. *Ethics and Information Technology*, 3(1):13–19.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Florian Tramer and Dan Boneh. 2021. Differentially private learning needs better features (or much more data). In *International Conference on Learning Representations*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Xuan-Son Vu, Thanh-Son Nguyen, Duc-Trong Le, and Lili Jiang. 2020. Multimodal review generation with privacy and fairness awareness. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 414–425, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In the Proceedings of ICLR.

Hui Wang, Hanbin Zhao, Xi Li, and Xu Tan. 2018a. Progressive blockwise knowledge distillation for neural network acceleration. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 2769–2775.

Teng Wang, Xuefeng Zhang, Jingyu Feng, and Xinyu Yang. 2020a. A comprehensive survey on local differential privacy toward data statistics and analysis. *Sensors*, 20(24):7030.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020b. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems*, 33:5776–5788.

Wenjie Wang, Pengfei Tang, Jian Lou, and Li Xiong. 2021. Certified robustness to word substitution attack with differential privacy. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1102–1112, Online. Association for Computational Linguistics.

Xiaojie Wang, Rui Zhang, Yu Sun, and Jianzhong Qi. 2018b. Kdgan: Knowledge distillation with generative adversarial networks. *Advances in neural information processing systems*, 31.

Stanley L Warner. 1965. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69.

Benjamin Weggenmann and Florian Kerschbaum. 2018. Syntf: Synthetic and differentially private term frequency vectors for privacy-preserving text mining. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 305–314.

Benjamin Weggenmann, Valentin Rublack, Michael Andrejczuk, Justus Mattern, and Florian Kerschbaum. 2022. DP-VAE: Human-Readable Text Anonymization for Online Reviews with Differentially Private Variational Autoencoders. In *Proceedings of the ACM Web Conference 2022*, pages 721–731, Virtual Event. ACM.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *Transactions on Machine Learning Research*.

Paul J Werbos. 1988. Generalization of backpropagation with application to a recurrent gas market model. *Neural networks*, 1(4):339–356.

Alan F Westin. 1968. Privacy and freedom. *Washington and Lee Law Review*, 25(1):166.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. 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 1112–1122. Association for Computational Linguistics.

Oliver Williams and Frank McSherry. 2010. Probabilistic inference and differential privacy. *Advances in neural information processing systems*, 23.

Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. 2016. Quantized convolutional neural networks for mobile devices. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4820–4828. IEEE.

Xinwei Wu, Li Gong, and Deyi Xiong. 2022. Adaptive differential privacy for language model training. In *Proceedings of the First Workshop on Federated Learning for Natural Language Processing (FL4NLP 2022)*, pages 21–26, Dublin, Ireland. Association for Computational Linguistics.

Shangyu Xie and Yuan Hong. 2022. Differentially private instance encoding against privacy attacks. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Student Research Workshop*, pages 172–180, Hybrid: Seattle, Washington + Online. Association for Computational Linguistics.

Chang Xu, Jun Wang, Francisco Guzmán, Benjamin Rubinstein, and Trevor Cohn. 2021a. Mitigating data poisoning in text classification with differential privacy. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages

4348–4356, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jiacheng Xu, Zhe Gan, Yu Cheng, and Jingjing Liu. 2020a. Discourse-aware neural extractive text summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5021–5031, Online. Association for Computational Linguistics.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks? In *Proceedings of International Conference on Learning Representations (ICLR 2019)*, pages 1–17, New Orleans, Louisiana.

Nan Xu, Oluwaseyi Feyisetan, Abhinav Aggarwal, Zekun Xu, and Nathanael Teissier. 2021b. Density-aware differentially private textual perturbations using truncated gumbel noise. In *The International FLAIRS Conference Proceedings*, volume 34.

Zekun Xu, Abhinav Aggarwal, Oluwaseyi Feyisetan, and Nathanael Teissier. 2020b. A differentially private text perturbation method using regularized mahalanobis metric. In *Proceedings of the Second Workshop on Privacy in NLP*, pages 7–17, Online. Association for Computational Linguistics.

Zekun Xu, Abhinav Aggarwal, Oluwaseyi Feyisetan, and Nathanael Teissier. 2021c. On a utilitarian approach to privacy preserving text generation. In *Proceedings of the Third Workshop on Privacy in Natural Language Processing*, pages 11–20, Online. Association for Computational Linguistics.

Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2016. Revisiting Semi-Supervised Learning with Graph Embeddings. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48, pages 40–48, New York, NY, USA. PMLR.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. 2017. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7130–7138. IEEE.

Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. 2016. Neural generative question answering. In *Proceedings of the Workshop on Human-Computer Question Answering*, pages 36–42.

Ying Yin and Ivan Habernal. 2022. Privacy-preserving models for legal natural language processing. In *Proceedings of the Natural Legal Language Processing*

Workshop 2022, pages 172–183, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Ghosh, Akash Bharadwaj, Jessica Zhao, Graham Cormode, and Ilya Mironov. 2021. Opacus: User-Friendly Differential Privacy Library in PyTorch. *arXiv preprint*.

Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, et al. 2022. Differentially private fine-tuning of language models. In *International Conference on Learning Representations*.

Da Yu, Huishuai Zhang, Wei Chen, and Tie-Yan Liu. 2021. Do not let privacy overbill utility: Gradient embedding perturbation for private learning. In *International Conference on Learning Representations*.

Xiang Yue, Minxin Du, Tianhao Wang, Yaliang Li, Huan Sun, and Sherman SM Chow. 2021. Differential privacy for text analytics via natural text sanitization. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3853–3866.

Amir Zeldes. 2017. The GUM corpus: creating multilayer resources in the classroom. *Language Resources and Evaluation*, 51(3):581–612.

Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. 2019a. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3712–3721. IEEE Computer Society.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019b. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. 2018. Deep mutual learning. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4320–4328. IEEE.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and William B Dolan. 2020. Dialogpt: Large-scale generative pre-training for conversational response generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 270–278.

Zhuo Zhang, Xiangjing Hu, Lizhen Qu, Qifan Wang, and Zenglin Xu. 2022. Federated model decomposition with private vocabulary for text classification. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6413–6425.

Chen Zheng and Parisa Kordjamshidi. 2020. SRLGRN: Semantic role labeling graph reasoning network. In *Proceedings of the 2020 Conference on Empirical Methods*

in Natural Language Processing (EMNLP), pages 8881–8891, Online. Association for Computational Linguistics.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, pages 19–27.

Alexander Ziller, Dmitrii Usynin, Rickmer Braren, Marcus Makowski, Daniel Rueckert, and Georgios Kaissis. 2021. Medical imaging deep learning with differential privacy. *Scientific Reports*, 11(1):1–8.