Knowledge Graphs and Graph Neural Networks for Semantic Parsing

Vom Fachbereich Informatik
der Technischen Universität Darmstadt
genehmigte

Dissertation

zur Erlangung des akademischen Grades Dr.-Ing.

vorgelegt von
Daniil Sorokin
geboren in Otradnoje, Russland

Tag der Einreichung: 17. April 2021
Tag der Disputation: 10. Juni 2021

Referenten: Prof. Dr. Iryna Gurevych, Darmstadt
Prof. Dr. Kristian Kersting, Darmstadt
Prof. Dan Roth, Ph.D., Pennsylvania, USA

Darmstadt 2021
D17

Sorokin, Daniil:
Knowledge Graphs and Graph Neural Networks for Semantic Parsing
Darmstadt, Technische Universität Darmstadt
Year thesis published in TUprints: 2021
Day of the viva voce: 10. Juni 2021
URN: urn:nbn:de:tuda-tuprints-191875
URL: http://tuprints.ulb.tu-darmstadt.de/19187

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

Ehrenwörtliche Erklärung[1]

Hiermit erkläre ich, die vorgelegte Arbeit zur Erlangung des akademischen Grades "Dr.-Ing." mit dem Titel "Knowledge Graphs and Graph Neural Networks for Semantic Parsing" selbständig und ausschließlich unter Verwendung der angegebenen Hilfsmittel erstellt zu haben. Ich habe bisher noch keinen Promotionsversuch unternommen.

Berlin, den 18. August 2021

Daniil Sorokin

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

Wissenschaftlicher Werdegang des Verfassers[2]

09/2007 – 07/2011 Bachelor of Arts an der Staatlichen Universität St. Petersburg

10/2011 – 01/2014 Master of Arts an der Eberhard Karls Universität Tübingen

04/2015 – 06/2019 Doktorand am Fachgebiet Ubiquitous Knowledge Processing (UKP-Lab) der Technischen Universität Darmstadt

08/2019 – heute Applied Scientist bei Amazon Deutschland

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

Anmerkungen zum Umgang mit Forschungsdaten

Gemäß der "Leitlinien zum Umgang mit Forschungsdaten" der Deutschen Forschungsgemeinschaft[3] wurden alle im Zusammenhang mit dieser Dissertation entstandenen Forschungsdaten langfristig archiviert und sofern möglich öffentlich zugänglich gemacht. Folgende Forschungsdaten wurden frei verfügbar gemacht:

- Software
  - Die für die in Abschnitt 3.1 beschriebenen Experimente notwendige Software steht unter der Apache-Lizenz 2.0 unter `https://github.com/UKPLab/emnlp2017-relation-extraction/` zur Verfügung.
  - Die in Abschnitt 3.2 beschriebene VCG und Simple VCG Systeme stehen unter der Apache-Lizenz 2.0 unter `https://github.com/UKPLab/starsem2018-entity-linking` zur Verfügung.
  - Die für die in Abschnitt 3.3.1 beschriebenen Experimente notwendige Software steht unter der Apache-Lizenz 2.0 unter `https://github.com/UKPLab/emnlp2018-argmin-commonsense-knowledge` zur Verfügung.
  - Das in Abschnitt 3.3.2 beschriebene Athene-System für den FEVER Shared Task steht unter der Apache-Lizenz 2.0 unter `https://github.com/UKPLab/fever-2018-team-athene` zur Verfügung.
  - Die für die in Kapitel 4 beschriebenen Experimente notwendige Software steht unter der Apache-Lizenz 2.0 unter `https://github.com/UKPLab/coling2018-graph-neural-networks-question-answering` zur Verfügung.
  - Das in Kapitel 5 beschriebene web-basierte System steht unter der Apache-Lizenz 2.0 unter `https://github.com/UKPLab/emnlp2018-question-answering-interface` zur Verfügung.
- Korpora
  - Das in Abschnitt 3.1.2 beschriebene, neu erstellte Korpus steht unter der Creative Commons-BY-SA-4.0-Lizenz `https://public.ukp.informatik.tu-darmstadt.de/emnlp2017-relation-extraction/` zur Verfügung.
- Forschungsergebnisse
  - Alle im Zusammenhang mit dieser Dissertation stehenden Publikationen sind in der ACL Anthology (`http://aclanthology.info/`) verfügbar.

[3] `http://dfg.de/download/pdf/foerderung/antragstellung/forschungsdaten/richtlinien_forschungsdaten.pdf`

– Alle Forschungsergebnisse sind zudem auch in dieser Dissertation selbst dokumentiert, die von der Universitäts- und Landesbibliothek Darmstadt zur Verfügung gestellt wird.

Abstract

Human communication is inevitably grounded in the real world. Existing work on natural language processing uses structured knowledge bases to ground language expressions. The process of linking entities and relations in a text to world knowledge and composing them into a single coherent structure constitutes a semantic parsing problem. The output of a semantic parser is a grounded semantic representation of a text, which can be universally used for downstream applications, such as fact checking or question answering.

This dissertation is concerned with improving the accuracy of grounding methods and with incorporating the grounding of individual elements and the construction of the full structured representation into one unified method. We present three main contributions: (1) we develop new methods to link texts to a knowledge base that integrate context information; (2) we introduce Graph Neural Networks for encoding structured semantic representations; (3) we explore generalization potential of the developed knowledge-based methods and apply them on natural language understanding tasks.

For our first contribution, we investigate two tasks that focus on linking elements of a text to external knowledge: relation extraction and entity linking. Relation extraction identifies relations in a text and classifies them into one of the types in a knowledge base schema. Traditionally, relations in a sentence are processed one-by-one. Instead, we propose an approach that considers multiple relations simultaneously and improves upon the previous work. The goal of entity linking is to find and disambiguate entity mentions in a text. A knowledge base contains millions of world entities, which span different categories from common concepts to famous people and place names. We present a new architecture for entity linking that is effective across diverse entity categories.

Our second contribution is centered on a grounded semantic parser. Previous semantic parsing methods grounded individual elements in isolation and composed them later into a complete semantic representation. Such approaches struggle with semantic representations that include multiple grounded elements, world entities and semantic relations. We integrate the grounding step and the construction of a full semantic representation into a single architecture. To encode semantic representations, we adapt Gated Graph Neural Networks for this task for the first time. Our semantic parsing methods are less prone to error propagation and are more robust for constructing semantic representations with multiple relations. We prove the efficiency of our grounded semantic parser empirically on the challenging open-domain question answering task.

In our third contribution, we cover the extrinsic evaluation of the developed methods on three applications: argumentative reasoning, fact verification and text comprehension. We show that our methods can be successfully transferred to other tasks and datasets that they were not trained on.

Zusammenfassung

Die menschliche Kommunikation ist unmittelbar in der realen Welt verankert. Bestehende Methoden zur Verarbeitung natürlicher Sprachen verwenden strukturierte Wissensdatenbanken, um sprachliche Ausdrücke zu verankern. Die Aufgabe des semantischen Parsing besteht darin, Entitäten und Relationen in einem Text mit dem Weltwissen zu verknüpfen, das in einer Wissensdatenbank kodiert ist. Als Output generiert ein semantischer Parser eine universelle semantische Textrepräsentation, die für Aufgaben wie Faktencheck oder Beantwortung von Fragen eingesetzt werden kann.

In dieser Dissertation wird untersucht, wie man die Genauigkeit des semantischen Parsers verbessert, indem man einzelne Textelemente mit dem Weltwissen verknüpft und strukturierte semantische Repräsentationen in einer einheitlichen Architektur konstruiert. Wir präsentieren drei Hauptbeiträge: (1) Wir entwickeln neue kontextsensitive Methoden, um Texte mit der Wissensdatenbank zu verlinken. (2) Wir führen Graph Neural Networks zum Codieren strukturierter semantischen Repräsentationen ein. (3) Wir untersuchen das Generalisierungspotential der entwickelten wissensbasierten Methoden und wenden sie auf Aufgaben an, die ein tiefes Sprachverständnis verlangen.

In unserem ersten Beitrag untersuchen wir zwei Aufgaben, die sich auf die Verknüpfung von Textelementen mit externem Wissen konzentrieren: Relationsextraktion und Entitätsverknüpfung. Die Relationsextraktion identifiziert Relationen in einem Text und ordnet sie einem der Typen des Wissensdatenbankschemas zu. Traditionell werden Relationen in einem Satz einzeln verarbeitet. Stattdessen schlagen wir einen neuen Ansatz vor, der mehrere Relationen gleichzeitig berücksichtigt. Unser System liefert bessere Ergebnisse als existierende Ansätze. Das Ziel der Entitätsverknüpfung besteht darin, Entitätserwähnungen in einem Text zu finden und zu disambiguieren. Eine Wissensdatenbank enthält Millionen von Entitäten, die verschiedene Kategorien von Ortsnamen bis hin zu berühmten Personen umfassen. Wir präsentieren eine neue Architektur für die Entitätsverknüpfung, die über verschiedene Entitätskategorien hinweg wirksam ist.

Unser zweiter Beitrag konzentriert sich auf einen semantischen Parser, der eine mit der Wissensdatenbank verknüpfte semantische Repräsentation eines Textes produziert. Frühere Methoden haben einzelne Textelemente (Entitäten und Relationen) isoliert verlinkt und sie erst später zu einer vollständigen semantischen Repräsentation zusammengesetzt. Solche Ansätze schneiden jedoch schlecht ab, wenn eine semantische Repräsentation mehrere Entitäten und Relationen enthält. Wir integrieren den Verknüpfungsschritt und die Konstruktion einer semantischen Repräsentation in eine gemeinsame Architektur. Um semantische Repräsentationen zu codieren, setzen wir zum ersten Mal Gated Graph Neural Networks für diese Aufgabe ein. Unsere Parsing-Methoden sind weniger anfällig für Fehlerfortpflanzung und robuster für die Erstellung semantischer Repräsentationen mit mehreren Beziehungen. Wir beweisen die Effizienz unseres semantischen

Parsers empirisch anhand eines Datensatzes zur Fragenbeantwortung.

In unserem dritten Beitrag behandeln wir die extrinsische Evaluation der entwickelten Methoden an drei Aufgaben: argumentatives Reasoning, Faktencheck und Text Comprehension. Wir zeigen, dass unsere Methoden erfolgreich auf andere Aufgaben und Datensätze übertragen werden können, für die sie nicht trainiert wurden.

Acknowledgments

I would like to express my warmest gratitude to all people who supported me on this journey.

It was my great fortune to have conducted this research at the Ubiquitous Knowledge Processing Lab under the supervision of Prof. Dr. Iryna Gurevych. I am grateful to her for creating this unique and successful academic community and I am honoured to have been accepted into it. I am particularly thankful to Prof. Gurevych for challenging me to do more and to aim higher. Further, I would like to thank Prof. Dr. Kristian Kersting and Prof. Dr. Dan Roth for serving as reviewers for this thesis.

Thanks are also due to my colleagues at UKP with whom I had a chance to work closely throughout these years: Dr. Teresa Botschen, Ilia Kuznetsov, Dr. Andreas Hanselowski, Dr. Maxime Peyrard, Dr. Judith Eckle-Kohler, Dr. Claudia Schulz, Dr. Tristan Miller, Dr. Lisa Beinborn, Dr. Silvana Hartmann, Dr. Emily Jamison, Dr. Hatem Mousselly-Sergieh, Michael Bugert and Dr. Christian Meyer. And special thanks to Dr. Teresa Botschen, Ilia Kuznetsov, Dr. Christian Meyer, Dr. Nafise Sadat Moosavi, Andreas Rücklé, Dr. Ivan Habernal, Gisela Vallejo and Yevgeniy Puzikov who have found time to peer-review parts of this thesis. I am also indebted to the co-authors of the papers upon which some of this dissertation is based. I thank Raktim Bora, Moudud Hassan, Lars Wolf, Daniel Faber, Patricia Heidt, Anadi Tyagi, and Marco Huber. It was a delightful and learning experience to (co-)supervise your Bachelor or Master theses.

It is impossible to enumerate all insightful discussions, entertaining talks, engaging workshops and joyful lunch pauses that we have had together. I thank my colleagues at UKP and in the AIPHES graduate school: Dr. Ana Marasović, Dr. Avinesh P.V.S., Dr. Tobias Falke, Dr. Teresa Botschen, Ilia Kuznetsov, Andreas Rücklé, Yevgeniy Puzikov for going through challenges together.

I would like to express my warm regards to everyone I have had the luck to meet over the past years at TU Darmstadt, Heidelberg University, numerous conferences, summer schools, and during my internship in London.

Finally, I am very thankful to my family and friends for all those moments when I could step aside from work and just be with them. Thank you for not always taking me seriously.

I have no words to describe how grateful I am to my wife Katja for keeping me afloat all this time. You were always there to help.

Contents

Chapter 1

Introduction

1.1 Motivation

Natural language is the ultimate tool to transfer and store knowledge in a compressed and efficient form. Human encoding of the information in the natural language form relies on the commonly shared world knowledge and commonsense understanding, on the situation awareness and on the shared geographical and temporal spaces (Bender and Koller, 2020). This ensures that redundant commonly known or already shared information is not repeated (Grice, 1975). Easily accessible to humans, the knowledge described as text is almost obscure to machines. The automatic processing of textual data has advanced dramatically in the last decades, yet it remains imprecise and struggles with the explicit representation of the meaning of the natural language. One of the major challenges for automatic processing is grounding the language in the shared world and common knowledge (Harnad, 1990; Bisk et al., 2020). Precise automatic processing of textual data needs to understand the meaning of a piece of human language, interpret it and put it into a broader context of the already available knowledge. Our main research direction in this dissertation is to use the world knowledge available in structured form to aid the automatic text processing. We develop a grounded semantic parser that links a natural language utterance to an external knowledge base and we tackle a particular language usage scenario: answering questions about facts.

Natural langauge processing (NLP) is a research field that studies automatic text processing and interpretation. NLP is concerned with different methods to process information in the natural language form so that it can be stored and interpreted by a machine. Storing information in machines as precisely and explicitly as possible has been studied by knowledge information and computer science researchers. Relational databases and structured knowledge bases are the most prominent data structures for machine-readable knowledge (Färber et al., 2015). In this dissertation, we are using existing structured world knowledge to disambiguate the language input and to represent it in a machine-readable and formally interpretable way.

To give an example, the meaning of the natural language expression on the left in Figure 1.1 could be encoded with a graphical machine-readable semantic representation on the right. The representation on the right uses unambiguous entities

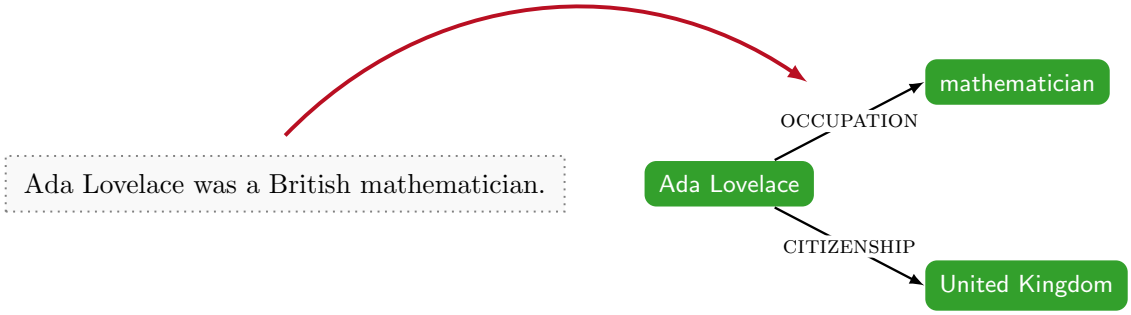

Figure 1.1: Encoding of the natural language utterance in a graphical machine-readable representation with relations and entities of an external knowledge base.

and relations from the schema of the open Wikidata knowledge base[1]. Through the fixed knowledge base schema, we can ensure that different ways of expressing the same information are encoded with the same relations and entities. Because Wikidata stores information about the world using the same schema, this graphical semantic representation is also interpretable: by comparing the individual relations in the representation with the reference information in the knowledge base, we can determine if the whole expression is true or false.

In Figure 1.1, the natural language expression is encoded with the semantic representation, which is a symbolic expression that uses a special semantic language based on the Wikidata schema. As a knowledge base, Wikidata provides a model of the world (or a domain), in which the symbolic expression is interpreted. Thereby, the original natural language expression is now linked to the world knowledge in the knowledge base through the graphical semantic representation. And the interpretation of the language expression is grounded in the external knowledge base. In this work, we assume that disambiguating the meaning of the input and linking it to the knowledge base model of the world constitutes *understanding* of the natural language.

The relation between a symbolic expression and a model of the world has been studied in logic (van Eijck and Unger, 2010). A logical proposition is expressed in terms of symbols that realize their meaning in a particular interpretation domain, which is our representation of the world. The whole proposition is evaluated in a compositional manner after the individual symbols are grounded. Montague (1974) proposed a compositional view on meaning of language: the individual symbols, words or morphemes, get their meaning only in the context of shared world knowledge, and the meaning of the whole natural language expression is derived in a compositional manner. The linguistics subfield of formal semantics has concerned itself with the rules of encoding the language into formalized logical meaning representations that follow the compositionality principle. NLP has naturally adopted the formal semantics analysis (Liang, 2016). As we saw

[1] `http://wikidata.org/`

in the above example in Figure 1.1, structured knowledge bases can serve as an interpretation domain to ground symbolic representations in a practical NLP setting.

Mapping the natural language input to formalized machine-readable semantic representations is ultimately the task of *semantic parsing*, one of the central tasks of NLP. A semantic parsing framework that allows the linking of utterances to machine-readable knowledge representations would ultimately lead to a universally accessible interface to the stored knowledge. Thus, the task of semantic parsing is concerned with the interpretation and disambiguation of natural language, with the extraction of world knowledge and with the generation of a semantic representation for the whole expression using a machine-readable unambiguous and precise semantic language. The same meaning in natural language, if expressed differently, should be linked to the same representation in terms of the unambiguous knowledge base elements. Once we can map a text to an unambiguous semantic representation grounded in world knowledge, we can transform it and further work with it in downstream applications.

As demonstrated in Figure 1.1, linking all elements in the text and extracting complex knowledge naturally becomes a semantic parsing problem (Parikh et al., 2015). And by linking the text and the external knowledge through a structured semantic representation, we enable a variety of NLP applications (Liang, 2016). For instance, related information about the world can be used to enrich the input text with additional information for reasoning. Or we can compare the semantic representation to the information contained in the knowledge base and thereby perform fact verification. If the input text is a question, we can retrieve the missing information from the knowledge base to answer it. Knowledge base question answering is one of the main applications for semantic parsing in NLP (Yao et al., 2014) and it is the primary task for semantic parsing, which we consider in this dissertation. We apply our methods further on fact verification, argumentative reasoning and text comprehension tasks to demonstrate the wide applicability of knowledge base semantic parsing.

NLP approaches that build explicit semantic representations are more broadly called structured approaches. An alternative NLP paradigm is to bypass storing the meaning of a text as an explicit representation and instead to build a system that solves the single target task directly. For instance, it can be feasible to factually verify a sentence by embedding it together with the relevant evidence in a text form in a statistical model and by predicting a binary true or false label. Such an approach would not construct an explicit structured semantic representation and could, as a result, be more straightforward. The semantic interpretation of the input exists in these methods only in a form of continuously learned numerical representations inside the statistical model (Sutskever et al., 2014). These methods are more broadly called continuous approaches (Goodfellow et al., 2016, Chapter 1). The critique of fully end-to-end continuous approaches that seek to avoid any explicit structure has highlighted the challenges they face in complex language

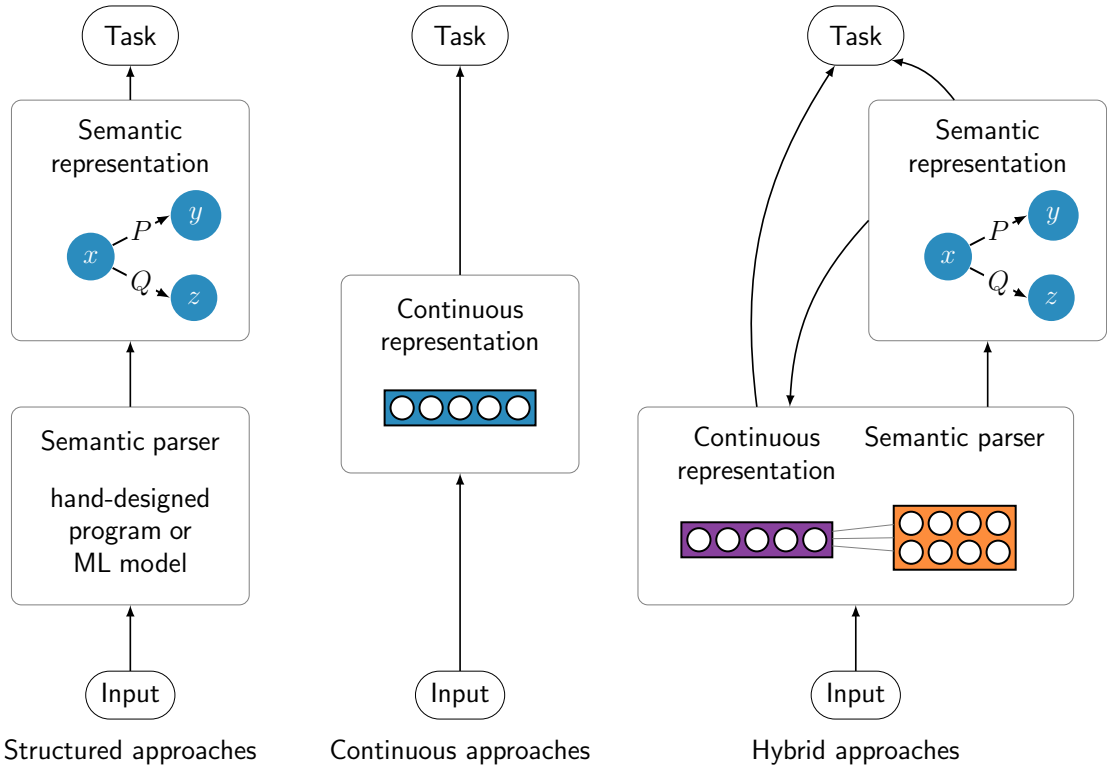

Figure 1.2: A comparison of structured, continuous and hybrid approaches for NLP tasks. Structured approaches construct an explicit representation of the input and continuous approaches learn a vector representation automatically from data. Hybrid approaches integrate structured representations with learning continuous representations and create a loop where a continuous representation may be updated from the structured semantic representation.

understanding and reasoning about structured data (Lake and Baroni, 2018). This has motivated researches to turn to hybrids of structure-based methods and continuous deep learning methods (Battaglia et al., 2018). In our work, we employ continuous vectors as intermediate encodings of text and of structured knowledge with the goal of combining the best of the two worlds into a hybrid approach for constructing explicit semantic representations. We visualize the difference between structured, continuous and hybrid approaches in Figure 1.2.

Grounded semantic parsing relies on linking the text to a knowledge base, but it can be challenging to process different types of information, such as a text and a knowledge base, together in a unified way. Previous work on linking a text to a knowledge base has been focusing on linking the entities and relations individually (see for instance, Francis-Landau et al., 2016 for entities and Zeng et al., 2015 for relations). In our work, we strive to improve the linking methods by incorporating context information and by disambiguating multiple entities and relations jointly. To construct the semantic representation, the existing methods have usually computed similarity between text fragments and elements of a knowledge base schema using hand-defined features and simple text encoding methods (Yih

et al., 2015; Reddy et al., 2017). Such approaches require time-consuming feature engineering and introduce error propagation. As we show in the later chapters, it causes semantic parsing models to struggle with semantic representations that include multiple entities and relations. In contrast, we create a semantic parsing model that encodes the text and the external knowledge into continuous vectors in a unified way, which improves the construction of semantic representations. In particular, we model semantic representations and knowledge base elements as graphs and we introduce a machine learning framework of Graph Neural Networks (GNNs) (Scarselli et al., 2009) to directly encode all of the properties of a graph together into a vector.

The starting point for this thesis is the assumption that it is possible to disambiguate a natural language input to such an extent that it can be encoded in a structured machine-readable form. We take this to be true within the context of the NLP applications that we consider in this work and under the requirement that the interpretation domain is known in advance and is bounded by an existing knowledge base. We use the Wikidata knowledge base to define the interpretation domain in this work. Wikidata is a collaboratively constructed knowledge base with community quality control. It is well-connected with other resources: entries are linked to Wikipedia articles and are annotated with identifiers from other knowledge bases if available. Färber et al. (2015) compared Wikidata with other contemporary knowledge bases and found that it had the most features from the their knowledge base checklist[2]. For that reasons, we choose Wikidata as the interpretation domain, where the semantic representations are grounded.

Taken together, the goal of this dissertation is to provide new context-aware methods for linking a text to a knowledge base and to build upon them to create a unified grounded semantic parsing model for NLP applications. We break down this goal into concrete research questions in the next section.

1.2 Research Questions

This thesis addresses the following three research questions:

RQ1 What are the challenges for linking a text to an external knowledge base and how can the linking methods be improved?
To interpret a text in the context of the external knowledge and to use the resulting interpretation for a downstream NLP application we need to perform a set of conceptual tasks. We rely on a logic-based semantics approach to formalize the interpretation process in terms of constructing an explicit meaning representation and to break it down into individual steps. Once the interpretation steps are set, we focus on those that involve disambiguation and linking of individual components and are thus central to the process of connecting a text to the external knowledge.

[2] See Table 8 in Färber et al. (2015), Wikidata has since then also added an official SPARQL endpoint (the checklist item 5.).

We address the following two aspects: what are the state-of-the-art methods for linking of individual components (relations and entities) to the knowledge base? And what are the main weaknesses of the existing methods for knowledge base linking and how they can be improved?

RQ2 How to encode structured semantic representations for effective grounded semantic parsing?
After we improve the steps that are needed to link a text to a knowledge base, we turn to the problem how to construct semantic representations and how such structured knowledge representations can be effectively processed. We investigate machine learning methods to encode the text and the corresponding semantic representations jointly. The knowledge base is stored as a graph and the corresponding semantic representations are graph structures, as well. Hence, we dive deep on the application of GNNs to this problem. Successfully encoding a structured semantic representation into a vector enables us to select the correct representation for the input text and is crucial for semantic parsing. We use the knowledge base question answering task as the main application to motivate and evaluate our methods.

RQ3 Can the knowledge base linking methods and the grounded semantic parser generalize to new data and be embedded into natural language understanding applications?
Linking methods for the individual text elements and the methods for semantic parsing have a potential to enrich NLP applications with external knowledge. Within this research question, we perform extrinsic evaluation for the developed linking and semantic parsing methods and show their ability to generalize beyond the question answering task. First, we explore fact verification and argument comprehension architectures that rely on the linking of individual entity mentions in a text. Second, we apply the semantic parsing model to the text comprehension task. And finally, we describe an interactive evaluation tool for the linking and semantic parsing components that we developed. The interactive tool provides insight into how the text interpretation with linking text elements to the knowledge base and with constructing a full graphical semantic representation works on a user input text.

1.3 Contributions

To answer **RQ1**, we present **new methods to link texts to a knowledge base and to compose information into a structured representation**.

- We start by examining the individual steps of the knowledge base semantic parsing task. In Chapter 2, we discuss logic-based grounded semantic representations and identify the conceptual steps that are needed to construct them. Most importantly, we single out the identification and extraction of knowledge base relations in text and linking of individual entity mentions as the main disambiguation and linking steps in the process. We show that ex-

isting methods for relation extraction and entity linking lack comprehensive modeling of the surrounding context.

- After dividing the process into individual steps, we first present a state-of-the-art **system for relation extraction** that considers multiple relations per sentence (the context-aware model in Section 3.1). We create a new **English Wikidata relation extraction dataset** to test the proposed method.
- Then we introduce our **system for the entity linking** problem. We particularly focus on linking entities across different entity categories to provide a basis for semantic parsing with a broad topic coverage (Section 3.2). The presented **Variable Context Granularity (VCG) model** encodes all entity features and shows the strongest performance on short and noisy web data. We use the developed model for entity linking in the English question answering task in Chapter 4 and therefore treat question answering as one of the main motivating applications for the entity linker.
- Our main **finding** is that rich context representations (our context-aware model for relation extraction and the VCG model for entity linking) are necessary for the successful linking of text elements to the knowledge base.

To answer **RQ2**, we present **graph-based methods to encode structured semantic representations**.

- We describe a semantic parsing approach to knowledge base question answering that builds upon our findings from relation extraction and entity linking and solves the task by **unifying the relation extraction and structured representation construction** (Chapter 4).
- We adapt GNNs to encode structured knowledge representations. In particular, we add a mechanism to improve the encoding of the relation labels in graphs. In Chapter 4, we present **a Graph Neural Network based semantic parsing model**, Gated Graph Neural Network on expanded graphs (ExpG-GNN), which is the state-of-the-art method for knowledge base question answering with explicit semantic representations (at the time of the first publication in Sorokin and Gurevych, 2018a). To evaluate our method, we convert an existing English question answering dataset to Wikidata.
- Our main **finding** is that GNN based methods have the best ability to encode rich semantic representations that have a graphical structure compared to non-graph methods.

To answer **RQ3**, we present **applications of the developed knowledge-based methods on natural language understanding tasks**.

- We provide **extrinsic evaluation** for our entity linking model and for the GNN based semantic parser. We show that external world knowledge can be used to improve the results on a diverse set of NLP tasks: text comprehension, argumentative reasoning and fact verification (Chapter 3 and

Chapter 4).

- We describe the system architecture of a **demo interface for instance-based analysis** that enables a user to explore the linking and semantic parsing models that were developed in this work (Chapter 5).
- Our main **finding** is that the new approaches that we developed for linking a text to the knowledge base and for grounded semantic parsing can be transferred to other NLP tasks and datasets that they were not trained on. This proves that our methods are able to generalize and have a wide applicability.

We summarize our findings and explore **future directions** for semantic parsing grounded in a knowledge base (Chapter 6). In particular, we discuss the possibility to incorporate logic-based reasoning and presupposition verification into NLP applications.

1.4 Publication Record

Different parts of this thesis have been previously published in international peer-reviewed conference proceedings in the fields of natural language processing and semantic web. Parts of these publications have been reused in this thesis. In the following, we list the publications and link them to the respective chapters. All the listed publications have joint authorship, and except as specifically noted below, the material presented in this thesis should be assumed to be primarily the contribution of the present author. Further, we state whether verbatim quotes from the publications are to be expected.

- In *Context-Aware Representations for Knowledge Base Relation Extraction* (Sorokin and Gurevych, 2017a), published at the EMNLP conference, we presented a deep Recurrent Neural Network architecture that jointly recognizes a set of knowledge base relations in the context of an input sentence. The data to train the architecture was created using the distant supervision method. The experiments are presented in Section 3.1. Passages of this publication are quoted verbatim.
- In *Mixing Context Granularities for Improved Entity Linking on Question Answering Data across Entity Categories* (Sorokin and Gurevych, 2018b), published at the *SEM conference, we addressed the problem of linking entity mentions to an external knowledge base in the context of question answering. The suggested method uses the limited surrounding context more effectively and covers named entities as well as common entities and concepts. We present the approach and the empirical evaluation in Section 3.2. Passages of this publication are quoted verbatim.
- In *End-to-end Representation Learning for Question Answering with Weak Supervision* (Sorokin and Gurevych, 2017b), published in Semantic Web Chal-

lenges — 4th SemWebEval Challenge at the ESWC conference, we described an automatic factoid question answering system for the QALD-7 Shared-Task. At the core of this system, there is a deep neural network architecture for end-to-end processing of input questions and semantic representations. Our system has won the subtask dedicated to question answering on the Wikidata knowledge base. The ideas from this paper have led to the work in Chapter 4.

- In *Modeling Semantics with Gated Graph Neural Networks for Knowledge Base Question Answering* (Sorokin and Gurevych, 2018a), published at the COLING conference, we demonstrated that it is possible to adapt Gated Graph Neural Networks to process structured semantic representations in a unified way. The described method improves the results on an open-domain knowledge base question answering dataset. The architecture and the empirical results are presented in Section 4.4. Passages of this publication are quoted verbatim.

- In *Interactive Instance-based Evaluation of Knowledge Base Question Answering* (Sorokin and Gurevych, 2018c), published at the EMNLP conference as a demo, we presented an interactive interface for error analysis and debugging of knowledge base linking models and question answering systems. We describe the developed tool and the user experience evaluation in Chapter 5. Passages of this publication are quoted verbatim.

- In *Frame- and Entity-based Knowledge for Common-Sense Argumentative Reasoning* (Botschen et al., 2018b), published at the ArgMin workshop in collaboration with Teresa Botschen, we explored the potential of using external world knowledge through frame and entity linking to facilitate the construction of correct argumentative chains. Our contribution in this work is the application of the Wikidata entity linking system on the task data, the construction of knowledge base embeddings, the design and execution of the experiments and the error analysis. The results are discussed in Section 3.3.1. Passages of this publication are quoted verbatim.

- In *Multi-Sentence Textual Entailment for Claim Verification* (Hanselowski et al., 2018), published at the FEVER workshop in collaboration with other doctoral students at TU Darmstadt, we contributed an entity linking model that connects input claims to the knowledge base. This enables a more accurate retrieval of evidences for the input claim through the linked entities. We describe the implications of entity linking and knowledge bases for automatic fact verification in Section 3.3.2.

1.5 Thesis Organization

The structure of this dissertation is illustrated in the circular diagram in Figure 1.3. This diagram highlights the main topics discussed in each chapter and how they

fold into each other. We repeat the diagram at the start of each chapter. In the following, we give an overview of the content of the thesis chapters.

Chapter 2 provides an overview of the task of semantic parsing and of the processing of semantic representations. This chapter covers the background necessary to understand and contextualize the research presented in the thesis as a whole. In addition, the subsequent chapters provide further background on the topics they cover. We introduce the concept of a structured open-domain knowledge base that functions as a state of the world where the semantics is grounded. We define an interpretation domain and a formal language for grounded semantic representations. We relate the semantic representations grounded in a knowledge base to logic-based representations and to ungrounded semantic parsing. The knowledge base forms a huge directed graph and the semantic representations are essentially small subgraphs of this knowledge. This chapter presents a formulation of GNNs, a powerful graph processing machine learning architecture.

In **Chapter 3**, we discuss the two main building blocks of a grounded semantic parser: a relation extraction system and an entity linker. The relation extraction system and the entity linker connect the text to the knowledge base through relations and entities and enable the construction of grounded text representations. We present two novel methods that advance the state of the art for each of these tasks. In the last two sections of the chapter, we use the entity linking approach to enhance the argument reasoning system with the information about world entities and to support the evidence extraction in an automatic fact verification pipeline. We observe a positive improvement across the various semantic understanding tasks facilitated with our new knowledge base linking methods.

Chapter 4 outlines the task of knowledge base question answering, a major application of the grounded semantic parsing methods. We apply the techniques of the previous chapter in combination with GNNs to create a robust question answering system. We argue for a semantic parsing based approach to question answering. After introducing the evaluation framework and the datasets, we empirically demonstrate that a semantic parsing system supported by a GNN outperforms the competing approaches in a challenging open-domain question answering task. We analyze the errors and the bottlenecks of the top-performing configuration.

Section 4.5 treats a further application of knowledge base semantic parsing methods in NLP. We expand our question answering system to the task of text comprehension where a system should be able to retrieve information from a set of textual documents. We apply our semantic parsing method based on GNNs to build a knowledge base interpretation of the question and use the interpretation as a constraint on the answer candidates from a text comprehension system. We conclude that a text comprehension system benefits from the information about the world accessible in the knowledge base.

To further understand the challenges of the task, we use a sequence-to-sequence

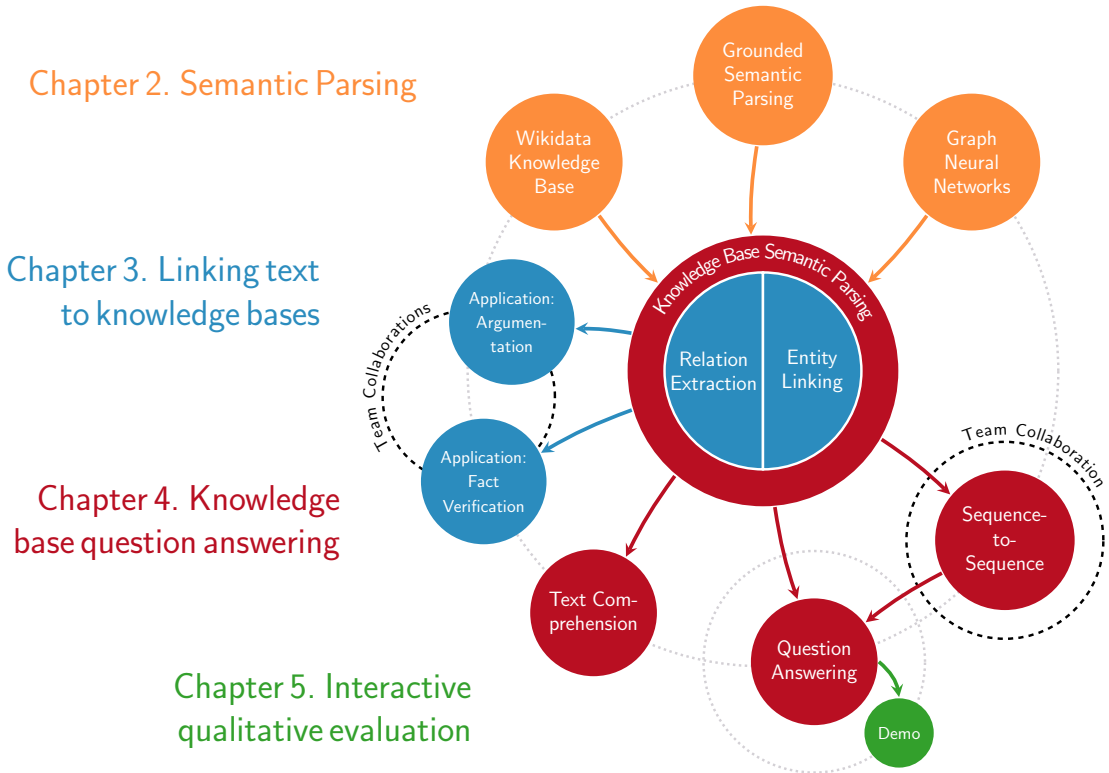

Figure 1.3: A schematic representation of the main topics of the thesis and their connections as a thesis diagram. Upper orange circles (Chapter 2): theoretical and methodological background on grounded semantic parsing with Wikidata. Blue semicircles at the center (Chapter 3): linking the relations and entities in text to Wikidata with rich context encoding. The large red circle at the center and the red circles in the bottom (Chapter 4): combining the linking methods into a semantic parser and applying it to question answering. The green circle at the very bottom (Chapter 5): an interactive evaluation analysis tool for the semantic parsing approach to question answering.

model targeted at simple questions and evaluate it in the combination with the main architecture in Section 4.6.

In **Chapter 5**, we describe an interactive web-based analysis system for semantic parsing and knowledge base question answering. Our system incorporates the individual components described in the previous chapters and showcases a complete prototype build upon the research findings of this thesis. Furthermore, the web-based interface is built around the debugging and error analysis requirements for question answering systems. We document the implementation aspects of the system and discuss its usage for the manual error analysis.

Chapter 6 summarizes the main contributions and outlines future work.

Chapter 2

Semantic Parsing

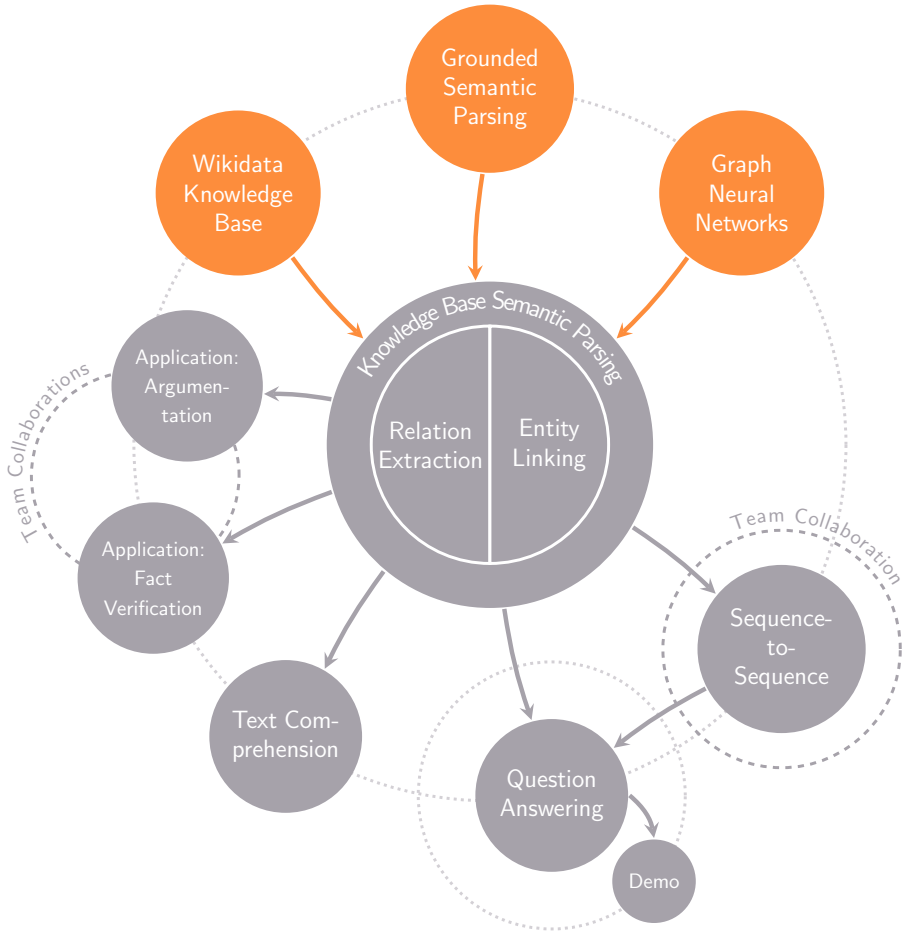

Figure 2.1: Chapter 2 in the thesis diagram. This chapter presents theoretical and methodological background on grounded semantic parsing with Wikidata and graph processing with neural networks (the three upper orange circles).

This chapter introduces the concepts and the previous work that are fundamental to the research presented in this thesis. We start by introducing the Wikidata knowledge base, which we will use to encode and interpret the meaning of the input text. We describe semantic parsing, the area of natural language processing that is concerned with interpretation and construction of explicit meaning representations. Finally, we present the Graph Neural Networks (GNNs), a processing framework that can be applied on knowledge base graphs and semantic representations.

We are foremost concerned with logic-based representations that combine the syntax of first-order logic and a pre-defined vocabulary of predicates and entities. There are two defining factors to our choice of the semantic representation that follow from the research goal of this work: we need to represent the linking of textual elements to a knowledge base and unify them into a single semantic representation of the input that can be used in NLP. In this chapter, we introduce semantic representations that employ external global knowledge base schema to set the vocabulary of the formal semantic language and we define the expressivity of this language in terms of first-order logic symbols.

We call the semantic representations *grounded* if they encode the meaning with a vocabulary of symbols that have an externally defined interpretation. We put the grounded representations into context by contrasting them with a subset of semantic representations that were introduced and used for NLP applications. The outlined semantic representation grounded in the Wikidata knowledge base is the core instrument that we use in our work. It encodes the linking between the text meaning and the information in the knowledge base.

This chapter lays the foundation for exploring the research questions that we have defined in Chapter 1. We provide the necessary background divided into three parts (see the upper part of Figure 2.1): (i) we introduce the Wikidata structured open-domain knowledge base that encodes information about the world using a strict predicate logic-conform schema (Section 2.1), (ii) we use the external knowledge to define grounded semantic representations and explore their relation to formal semantics and ungrounded semantic parsing (Section 2.2), (iii) we describe GNNs, a powerful machine learning architecture that is naturally suited for processing graphical semantic representations (Section 2.3).

2.1 The Wikidata Knowledge Base

In this section, we present the Wikidata knowledge base and contrast it with other sources of structured world knowledge. We motivate our choice of Wikidata as the primary resource that is used throughout all of the experiments in this work and explain the model of encoding the knowledge and meaning with the Wikidata schema.

As per our goal to link utterances with external knowledge and to use this linking to interpret the utterances' meaning, we are interested in resources that can be mapped to natural language and that offer a broader external interpretational context. Among previous work, the common choices of a knowledge base included Wikipedia[1], DBpedia[2], ConceptNet[3], YAGO[4] and Freebase (we refer to Färber et al., 2015 for a survey on knowledge bases).

[1] http://wikipedia.org
[2] http://wiki.dbpedia.org
[3] http://conceptnet.io
[4] http://yago-knowledge.org

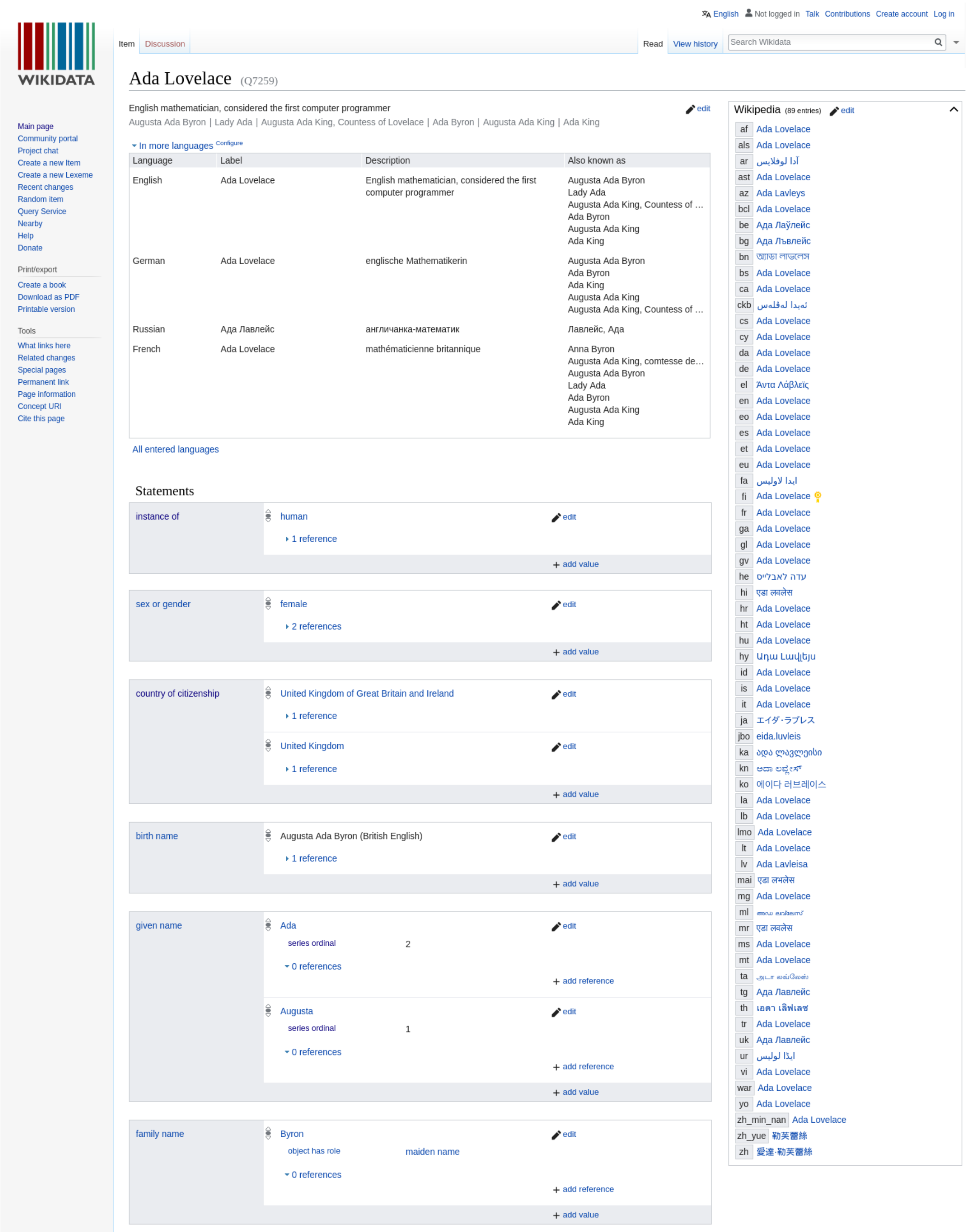

Figure 2.2: The web interface to the Wikidata knowledge base. The screenshot shows a Wikidata page for a single entity that represents Ada Lovelace, a programming pioneer, and the relations the entity has to other entities in Wikidata. The page also shows the links to the Wikipedia pages about this entity in various languages on the right side.

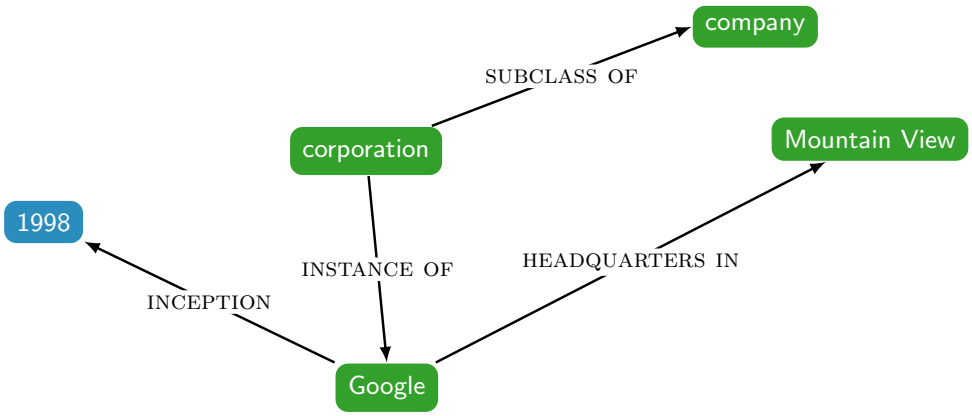

Figure 2.3: An example subgraph of Wikidata (only selected relations are shown). The entities (depicted here as green nodes) are connected with binary relations to each other or to values (depicted here as blue nodes), such as a date or a number.

We use the Wikidata open-domain knowledge base (Vrandečić and Krötzsch, 2014) throughout the experiments in this thesis. Wikidata is a general-purpose and ever-growing knowledge base that adheres to a strict information encoding schema. It encodes knowledge about world entities and relations between them. An entity and a binary relation type are the main building blocks of Wikidata. The knowledge base consists of individual relation instances or statements. A statement connects two entities with each other (⟨Darmstadt, country, Germany⟩), meaning 'Darmstadt is in Germany', or one entity with a value (⟨Darmstadt, population, 153,166⟩, meaning 'Darmstadt has a population of 153,166'. The collection of statements together forms a knowledge graph of entities and relations (see Figure 2.2 for the Wikidata web view for one entity and Figure 2.3 for a sample of entities and relations as a subgraph). We write a Wikidata statement with angel brackets and the relations are written in small capital letters: ⟨first entity, relation, second entity or value⟩.

Wikidata demonstrates several features that make it particularly attractive as an NLP resource. For instance, the entities in Wikidata directly correspond to the Wikipedia articles. This enables us to work with any data that was previously annotated for Wikipedia, DBpedia or other resources that also map to Wikipedia article names. Moreover, most entities in Wikidata have been also manually annotated with identifiers from other knowledge sources and databases, including, among others, the Freebase resource, which establishes a link between these two knowledge bases. We discuss some of the main Wikidata features in this section and further expand on the relevant properties in the subsequent chapters when they are used in the experiments.

Wikidata started in 2011 as a project of the Wikimedia foundation for a collaboratively constructed knowledge base that would unify the factual knowledge contained in Wikipedia. Wikidata has quickly expanded and was soon suggested as a most suitable alternative for a general-purpose knowledge base after the large

Freebase online graph was shut down in 2015[5]. Freebase was a common choice of a knowledge base in the previous work on semantic parsing (Berant et al., 2013; Yao et al., 2014) and we use it as point of comparison for Wikidata.

One of the most important Wikidata properties is the high quality of the data. It stems from the collaborative nature of the knowledge base and the established community rules, which set the notability criteria for including the information into the knowledge base and regulate the fact verification practices. Contributors discuss and verify all information before it is added to the knowledge base. This procedure ensures a higher data quality in Wikidata compared to other knowledge bases (Färber et al., 2015).

For comparison, the latest available Freebase dump consisted of more than 2 billion fact statements (relation instances), whereas Wikidata has only surpassed 70 million statements. One of the major reasons for the dramatic difference in the sizes of Wikidata and Freebase is the data policy of the former developed by the community. Although Wikidata, in the same way as Freebase, is designed to store the most of the common world knowledge, it intentionally avoids information pertaining to a specific domain. Freebase (as well as DBpedia and YAGO) was partially populated automatically without manual supervision and as a consequence contains a lot of superficial information that may exceed the *common* knowledge. For example, Freebase contains information on all FIFA sport events, from the world championship to local tournaments. Freebase was actively expanded with the goal to incorporate as much knowledge as possible and domain specific databases were merged into Freebase to increase its coverage (e.g. Music labels data). This strategy has resulted in a skewed distribution of facts (Tanon et al., 2016) and an uneven coverage of different knowledge domains, which makes it harder to develop NLP applications.

Effectively, one cannot use Freebase to approximately estimate if a fact is a part of the common knowledge. When it comes to sport events it might be recorded to the smallest detail, but the list of academic institutions might be incomplete for a particular country. In contrast to Freebase, Wikidata doesn't list all FIFA events, but includes only the most significant ones. It follows the same principles as Wikipedia on the notability of the entities and facts to be included[6]. A notable topic is essentially a topic that has gained significant attention by the world. As the result, the distribution of facts included in Wikidata is more uniform across the topics and for each topic the most notable aspects are described first. This is a useful property in practice: in information retrieval and other NLP applications, one can assume that events and entities included in Wikidata are also those that would be of relevance to the user first.

Given these observations, we consider Wikidata for our experiments on linking and grounding meaning in external knowledge. The continuous growth of

[5] Freebase is no longer officially supported: `https://developers.google.com/freebase/`

[6] `https://en.wikipedia.org/wiki/Wikipedia:Notability`

Relation type	Entity
country:P17	Darmstadt:Q2973
located in:P131	Hesse:Q1199
shares border:P47	Germany:Q183
instance of:P31	federal state:Q43702
subclass of:P279	EMNLP conference:Q18353514
sport:P641	athletics:Q542
citizenship:P27	Ada Lovelace:Q7259
spouse:P26	marriage:Q8445
founded by:P112	European Union:Q458
cast member:P161	Star Wars: The Force Awakens:Q6074

Table 2.1: An unordered selection of the common Wikidata relation types and entities.

Wikidata and its status as a community-managed knowledge base counterpart to Wikipedia facilitated its adoption in NLP in the recent years (Hewlett et al., 2016; Kaffee et al., 2018; Logan et al., 2019). At the moment, Wikidata contains more than 56 million entities and 2000 relation types.[7] To give an idea of the information stored in Wikidata, we list some common relation types and exemplary entities in Table 2.1. Unique IDs in Wikidata have a Q-prefix for entities and a P-prefix for relations. We omit the Wikidata IDs in some cases for clarity.

Formally, Wikidata can be represented as a large directed graph $W = (E, R, I)$, where E is a set of entities, R is a set of relation types and I is a collection of relation triples (statements): $\langle e_1, r, e_2 \rangle; r \in R; e_1, e_2 \in E$ (e. g. $\langle$Italy, capital, Rome$\rangle$). In practice, the data is stored in a graphical RDF format and the information can be retrieved with SPARQL queries.

Wikidata encodes all information uniformly using only binary relations. It defines a special instanceOf:P31 predicate to store the categorical information ($\langle$city(Rome)$\rangle \mapsto \langle$Rome, instanceOf, city$\rangle$) and n-ary relations are normally broken down into binary relations. For example, ternary relations in Wikidata are stored as three binary relation statements, which are interconnected through the entities: $\langle e_1, r_1, e_2 \rangle, \langle e_2, r_2, e_3 \rangle, \langle e_3, r_3, e_1 \rangle$. For instance, the fact that Carrie Fisher played Princess Leia in Star Wars is stored as $\langle$Carrie Fisher, character role, Princess Leia$\rangle$, $\langle$Carrie Fisher, cast member, Star Wars$\rangle$, $\langle$Princess Leia, present in work, Star Wars$\rangle$. In practice, such treatment of relations simplifies the implementation of NLP applications. We directly benefit from the streamlined Wikidata schema, when we define the corpus on relations in Section 3.1.2 and the semantic parser in Section 4.2. Wikidata relation types are in contrast to the Freebase schema that, in addition to binary relations, allows unary relations for categories (e.g. $\langle$city(Honolulu)$\rangle$) and includes special compound value type nodes, which

[7] `https://www.wikidata.org/wiki/Special:Statistics`

	Wikidata	Freebase
# of relation types	> 2500	> 25,000
# of entities	> 56M	> 47M
# of statements	> 70M	> 2000M
Schema	Only binary relations	Unary, binary and n-ary relations through special nodes
Entity categories	Yes, as INSTANCE_OF relation, categories are also entities	Yes, categories are unary relations
Reverse relations	Some relations have a reverse equivalent, but not always one-to-one	Reversed variants are created for almost every relation
Domains	No special treatment	Yes
Content sources	Manually added or verified	Manually, automatically from Wikipedia or licensed from other sources
Integration	Wikipedia articles map to Wikidata entities	Multiple existing annotated NLP datasets

Table 2.2: Summarized features of Wikidata in contrast to Freebase. The numbers of entities and relations are constantly changing, and we only include approximate values here (retrieved in May 2020).

are used to encode n-ary relations.

We summarize the comparison between Freebase, the common choice for a knowledge base in NLP applications up until its closure, and Wikidata in Table 2.2. In the next section, we describe how a knowledge base can be used to create structured semantic representations that are grounded and are interpretable in the model of the world encoded by the knowledge base.

2.2 Grounded Semantic Parsing

In this section, we describe grounded meaning representations for natural language utterances and the process of semantic parsing. We consider grounding semantic representations in the Wikidata knowledge base. With grounding, the meaning of an utterance is unambiguously encoded using the knowledge base schema. We start by surveying some of the available semantic representation languages and formalisms and then focus on the logic-based formal semantic representations, which will be the basis of our work as we move forward in the next chapters. The logic-based representations allow to integrate the knowledge

base schema into the semantic representation as a vocabulary of entities and predicates. Thereby, we encode the linking of the individual textual elements to the knowledge base and provide a unified semantic representation grounded in a knowledge base model of the world. In the last part of this section, we discuss how the logic-based representations grounded in a knowledge base can be derived for an input sentence.

2.2.1 Semantic Representations

The goal of semantic parsing is to map text to a complete and detailed meaning representation (Mooney, 2007). Mapping of natural language phrases to a formalized representation of their meaning has been attempted since the dawn of automated natural language processing (see, for example, Green et al., 1961). Arguably, representing the meaning of language in an unambiguous and machine-readable form is one of the main goals of NLP (Liang, 2016).

Many types of semantic representations were created in a search for a universal semantic formalism that can be used across NLP applications. For example, the discourse representation structure (Kamp and Reyle, 1996) or frame semantics (Fillmore, 1976) have been developed as general-purpose representations. At the same time, other semantic representations have been developed through the target application task that they meant to solve. For instance, database queries are used for unambiguous representation of natural language questions (Berant et al., 2013) or instructions for robots are mapped to a representation that is a sequence of domain-specific actions (Artzi and Zettlemoyer, 2013). In this section, we look at a sample of semantic formalisms without restricting the overview to a particular target task. Nevertheless, we only discuss the types of semantic representations that were used in practical NLP settings.

Our goal in this thesis is the development of the methods for parsing text into structured semantic representations and the advancement of the applications using those representations. Knowledge bases provide the grounding of such representations in the real world and the formal semantic language of the representations is based on the first-order logic. We lay down in this section why this combination is practically motivated. The complete enumeration of semantic formalisms is beyond the scope of our work, but we aim to set the scene and provide the background for our contributions on grounded semantic parsing. As we go along, we notice if other semantic representations can also be grounded in external knowledge and how expressive is the formal language of the representation.

A semantic representation uses a formal language to encode the meaning of a natural language utterance. The formal language is defined by its vocabulary and syntax. The vocabulary is a union of symbols that have external interpretation and of special operators (such as negation). The syntax describes how the symbols can be combined with each other and with the operators. Semantic languages vary with respect to their expressivity and formal interpretability. Some semantic

Formalism	Expressivity	Applications	Limitations
FrameNet Frame Semantics	A predefined fixed set of non-decomposable stereotyped event structures (frames), which can describe predicates, their arguments and relations between them.	Event retrieval, summarization through clustering the description of similar events.	Only events and participants can be encoded. No external grounding for frames is provided.
AMR	Conceptual tree structures, which use PropBank semantic roles for events and include logical operators for quantification, modals and negation.	Event retrieval, summarization through clustering the descriptions of similar events, logical inference.	No external grounding for PropBank roles is provided.
Natural language logic	Syntactic clauses without internal structure and inference relations between them. A minimum representation for basic reasoning.	Fact verification through the implication relation between clauses, summarization by proposition extraction.	Limited to propositional logic, no detailed semantic description below the clause level, no external grounding.
Propositional semantics	Syntactic dependency trees transformed to semantic propositions (a predicate with its arguments).	Proposition retrieval, summarization by proposition extraction.	Predicate-argument relations are limited to syntactic relations. No grounding.
Logic-based grounded semantics	Logic formulas that are constructed from pre-defined definitions for individual lexemes. Logic formulas can encode co-reference, quantification and negation and can be interpreted in an external world model.	Question answering on knowledge bases, reasoning and inference with logic rules, mapping instructions to actions.	Not a single semantic language, but a modeling approach. Most work is limited to a subset of first-order logic.

Table 2.3: A summary of the discussed semantic formalisms. The table includes five frequently used semantic representations, which exemplify major semantic language properties.

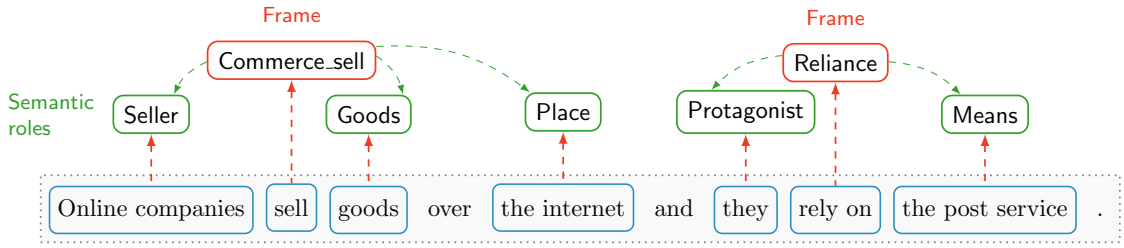

Figure 2.4: An example sentence with a frame-semantic representation using the FrameNet frames and semantic roles.

languages have simple syntax (e.g. frame semantics simply lists the frames for the given sentence) and some do not fix their vocabulary to a pre-defined interpretable set (e.g. propositional semantics uses any token spans from the input sentences directly to describe actions and their participants).

The vocabulary and the syntax of the semantic language influence the main properties of a semantic representation. We look, in particular, at the following two properties when we discuss the semantic representations: expressivity and formal interpretability. The expressivity describes what natural language phenomena can be reflected in a semantic representation. For instance, to encode anaphora in the semantic representation, the semantic language has to introduce variables or other means to refer to previously mentioned entities and events. Formal interpretability describes if it is possible to determine the truthfulness of the representation in an externally defined world model. It is an important property to enable logical inference with semantic representations.

Table 2.3 lists the five semantic formalisms that we examine in this section: Frame semantics (Fillmore, 1976), abstract meaning representations (AMR) (Banarescu et al., 2013), Natural language logic (Angeli and Manning, 2014), Propositional semantics (Stanovsky et al., 2016) and logic-based semantics (van Eijck and Unger, 2010). We have selected a representative sample of semantic representations ranging from a more expressive representation such as AMR (Banarescu et al., 2013) to representations with a strict vocabulary such as Frame semantics (Fillmore, 1976). This list is not exhaustive, but provides the context for a discussion of semantic representations used in NLP. We compare them against the logic-based grounded formalism that is the main semantic representation approach that we rely on in this work. Table 2.3 includes a note on the expressivity of the semantic formalism and the limitations, such as the lack of formal interpretability. We also name common NLP applications for each type of the semantic representation.

Frame semantics represents the meaning through predefined conceptual structures (frames) that characterize non-decomposable stereotyped events and actions (Fillmore, 1976). A frame is attached to frame evoking elements in the sentence, which can be one or several words. The attached frame encodes the core meaning of the frame evoking elements (e.g. the frame 'Commerce_sell' attached to the word 'sell' in Figure 2.4). Participants of the events encoded by the frames

are assigned semantic roles ('Online companies' and 'goods' in the example in Figure 2.4). Frame semantics operates with a fixed vocabulary of frames and semantic roles and describes the meaning in terms of these semantic primitives. The FrameNet resource is the largest collection of frame and semantic role definitions for English that follows the frame semantics formalism (Baker et al., 1998). The syntax of the frame-based representation is simple: frames act as n-ary predicates and the representation of a full sentence is a set of evoked frames with their participants.

Different expressions of the same event are mapped to the same frame (Das et al., 2014). In that regard, frame semantics can be used to disambiguate event mentions and relations to event participants. The expressivity of the frame semantics is defined by the vocabulary of frames and semantic roles. With a limited syntax, each new phenomenon has to have a dedicated entry in the vocabulary. For instance, the negation, which is not strictly an event on its own, was dedicated a special frame in FrameNet.[8] The frame-centric approach is tailored for event descriptions and is not well-suited to describe factual knowledge. The meaning aspects, such as casual relations or referential links, are lost when the representation is fixed as a set of frames. In Figure 2.4, the representation consists of the two identified frames with the corresponding frame evoking elements and the set of participants with the semantic roles, but no relations between the clauses are recognized and the co-reference relations are not reflected in the resulting semantic representation. There is no formal interpretability defined for the vocabulary of frames or roles, although in theory there are no barriers to provide it. The fixed vocabulary of frames would make it possible to extract reference knowledge from texts such as Wikipedia and store it as a model of the world. Then each new occurrence of the same frame could be judged against this frame-based world model. In practice, such a model would require a precise frame semantic parser or a hand-curated model of the true frame knowledge, which are not available right now.

AMR build upon the idea of PropBank semantic roles (Palmer et al., 2010) and extend them to a full semantic representation (Banarescu et al., 2013). PropBank style semantic roles are related to the frame semantics in that they also focus on predicate-argument structures. This type of representation was developed for the PropBank annotated corpus. The PropBank semantic roles are only meant to add a single semantic level for the predicate's arguments on top of the syntax level. These representations aim to abstract from concrete syntactic realizations of the predicate's arguments and combine word sense disambiguation for verbs with identification and labeling of its arguments. AMR uses predicates defined in PropBank with a first-order logic syntax that includes entity variables for co-reference, quantification and negation (see Figure 2.5). Polarity and modality are also encoded with special types of predicates. Overall, AMR offers an expressive representation that is, in the first place, limited by the PropBank vocabulary.

[8] `https://framenet2.icsi.berkeley.edu/fnReports/data/frameIndex.xml?frame=Negation`

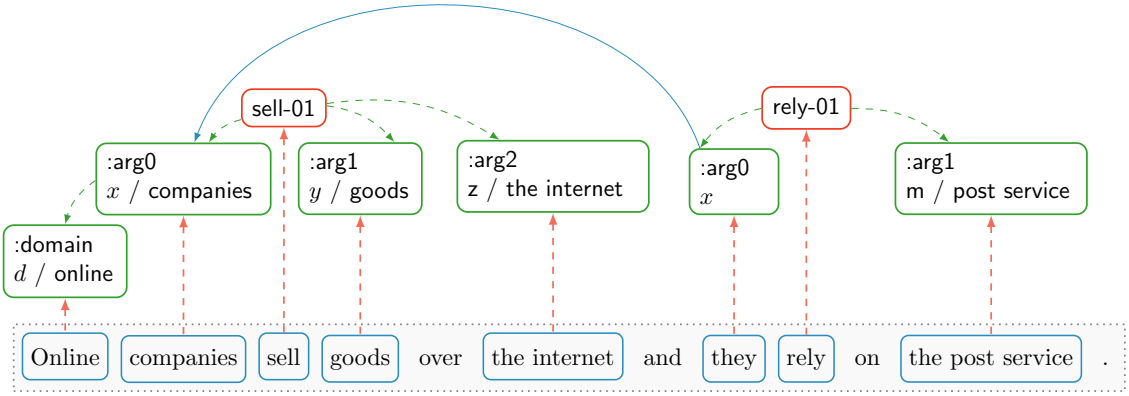

Figure 2.5: An example sentence with an AMR representation using the PropBank predicates and roles (graph notation). Note that this representation uses variables to establish a link between 'they' and 'Online companies'.

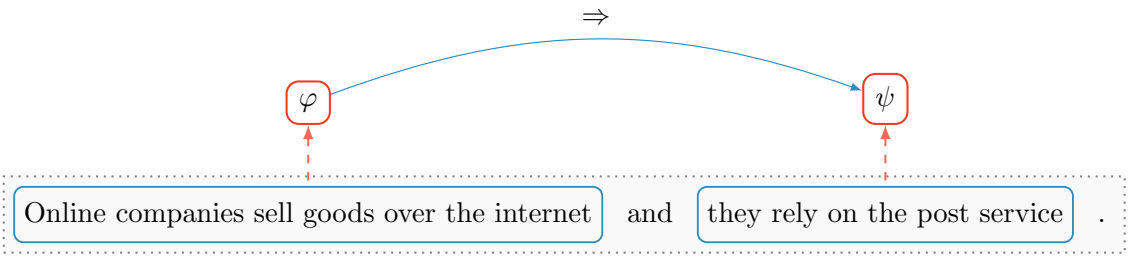

Figure 2.6: An example sentence with a natural language logic representation, which encodes the entailment relation (⇒) between the two propositions: 'the online companies sell goods over the internet' and hence have to 'rely on the post service'.

Through the first-order logic component, AMR supports some logical inference. The formal interpretability of AMR is undefined, but could be theoretically overcome in the same manner as for frame-semantic representations.

Natural language logic identifies text spans as propositions from utterances and only encodes the inference relations between a new statement and the previous statements (Angeli and Manning, 2014). The whole meaning representation is thereby a combination of the propositions and entailment relations between them (Figure 2.6). Hence it is not an attempt at a full meaning representation, but rather a representation sufficient for basic reasoning. In the follow-up work, Bowman et al. (2015) have operationalized a restricted set of three propositional relation types (ENTAILMENT, CONTRADICTION, NEUTRAL) to create the first large-scale corpus for recognizing natural language entailment. Although basically limited to propositional logic, this type of representation has proven effective for practical applications. For instance, automatic fact verification benefited a lot from treating the whole fact as a single proposition and learning a robust inference model (Thorne et al., 2018). This representation cannot be used to reconstruct meaning of individual sentences and it seems to be impossible to define formal

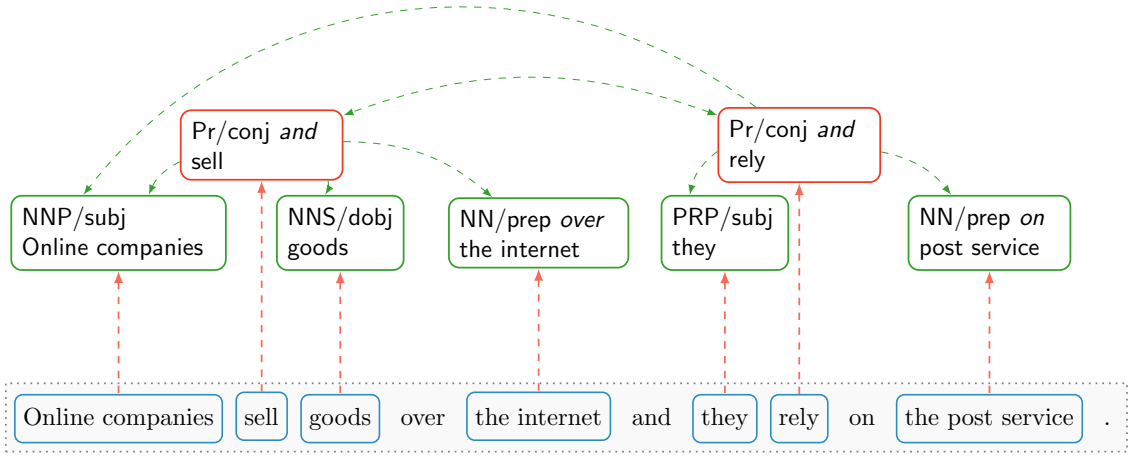

Figure 2.7: An example sentence represented with the propositional structures using the PropS parser. The nodes are annotated with their syntactic type/relation to the parent node.

interpretability on the proposition level.

Stanovsky et al. (2016) have presented another proposition-based approach that focuses on the inner structure of a proposition rather than the relations between them. They suggest to convert dependency trees through a series of deterministic transformations to a representation that simplifies the extraction of semantic phenomena that can be read from syntax. The focus of this type of representation is on the extraction of simple structured propositions centered around predicates, as opposed to natural language logic, which does not break down propositions into components. In that regard, the proposition extraction of Stanovsky et al. (2016) is more similar to frame structures that are directly linked to syntactic relations between words or to formal logic-based representations of Reddy et al. (2016). Essentially, Stanovsky et al. (2016) map the propositions with similar meaning to the same structural representations and mask out non-core syntactic details. Figure 2.7 shows the same example sentence we have used above annotated with the propositional structure using the PropS parser[9] of Stanovsky et al. (2016). The main advantage of this approach is the direct mapping to syntax. Such propositions are easy to extract and they offer a semantic representation layer that reduces certain syntactic variance. As the vocabulary of this representation is not fixed, it is not intended for formal interpretability.

Logic-based representations offer a broad framework for semantic representations that generally follows a simple idea: use first-order or higher-order logic as the syntax of the semantic representation and define a fixed vocabulary of primitive predicates and entities to cover the diverse language meaning. Montague (1974) pioneered the idea of using the unambiguous language of formal logic to describe the meaning of individual language elements. Different variations of the logic-

[9] `http://u.cs.biu.ac.il/~stanovg/props.html`

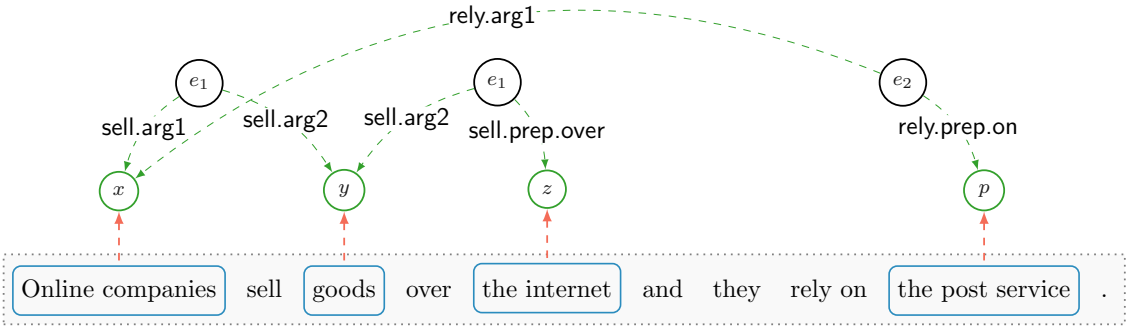

Figure 2.8: An example sentence represented with an event-based representation as defined in Reddy et al. (2016). Note that this representation establishes a direct link between the first arguments (arg1) of 'sell' and 'rely'.

based representations exist. Some settle on a simpler syntax of first-order logic and compensate the lack of expressivity through a large vocabulary of predicates. Others use the higher-order logic concepts to provide explicit descriptions for complex language phenomena such as modality (van Eijck and Unger, 2010). Most of the practical NLP applications of logic-based representations are based on the first-order logic with the focus on events (we refer to Reddy et al., 2014 as a prototypical example). Figure 2.8 shows a logic-based representation, where the event variables e_1 and e_2 are used to tie together the information that pertains to each event. We use only binary relations to connect entities and events in this example, following Reddy et al. (2014). The semantic representation in Figure 2.8 for the sentence "Online companies sell goods over the internet and they rely on the post service." can be rewritten as a logical formula:

$$\begin{aligned}\exists x y z p\, e_1 e_2 .\, &\text{OnlineCompanies}(x) \wedge \text{Goods}(y) \wedge \text{Internet}(z) \wedge \text{PostService}(p) \\ &\wedge \text{sell}(e_1) \wedge \text{arg1}(e_1, x) \wedge \text{arg2}(e_1, y) \wedge \text{prep.over}(e_1, z) \\ &\wedge \text{rely}(e_1) \wedge \text{arg1}(e_2, x) \wedge \text{prep.on}(e_2, p)\end{aligned} \tag{2.1}$$

The predicate-argument structures of first-order logic are a good match for knowledge base structures. As we have described above, the knowledge bases use entities and predicates of fixed arity to store the information. Therefore, a knowledge base can be used to define the vocabulary of a logic-based representation (Berant et al., 2013; Reddy et al., 2014). In that regard, the knowledge base serves as a model of the world that stores the truth, and the semantic representations can be interpreted against it. Apart from knowledge bases, the logic-based semantic language permits other types of grounding such as physical environments for robot actions (Artzi and Zettlemoyer, 2013).

The representations such as frame semantics or AMR already borrow the syntax from formal logic and can be seen as special cases of logic-based representations with a pre-defined vocabulary and limited syntax. Other representations, such as

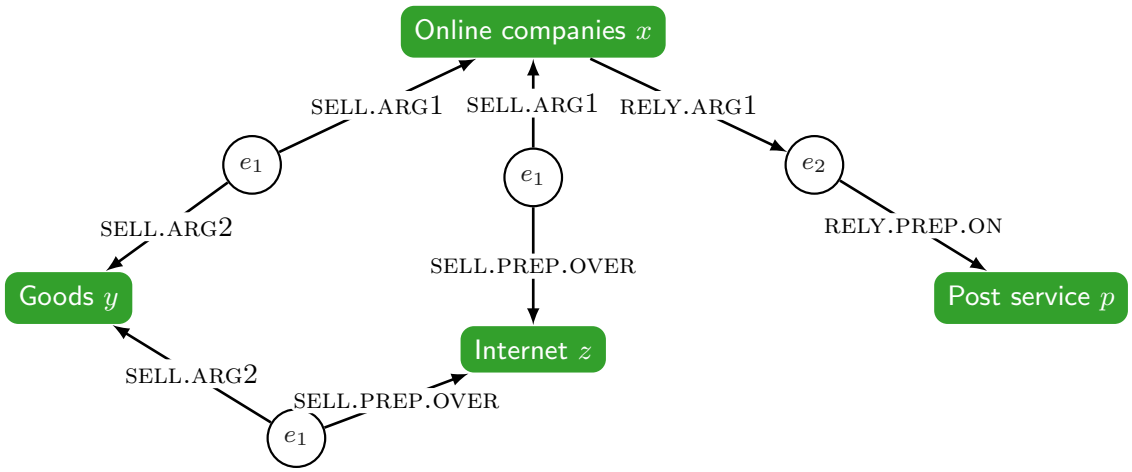

Figure 2.9: The logic-based representation from Figure 2.8 for the sentence "Online companies sell goods over the internet and they rely on the post service." visualized as a graph.

propositional semantics, focus on shallow semantic parsing instead of inference or formal interpretability. In this thesis, we are interested in enriching texts with knowledge-based methods. This means that we use a knowledge base as our model of the world and as a basis for formal interpretability of the semantic representations. Logic-based representations offer a framework for representing the text with an expressive semantic formalism that can be grounded in an external knowledge base and thereby linked to it. In the next section, we dive deeper into the logic-based semantic representations framework and how it can be grounded in an external knowledge base such as Wikidata.

2.2.2 Logic-based Grounded Representations

Our main goal is to offer a framework and methods for linking texts to a knowledge base and leverage the resulting linking for concrete NLP applications. As we have noted in the previous section, linking parts of the text to a knowledge base can be naturally incorporated into a logic-based representation. A logic-based semantic language with a vocabulary initialized from the knowledge base schema can be used to encode the result of linking the text in a structured form. Thus, we turn to logic-based grounded semantics as a way of encoding the result of linking the text to the knowledge base. We will see in the subsequent chapters, how having a structured grounded representation is beneficial for downstream NLP applications, such as question answering.

Let us have another look at the logic-based representation example in Figure 2.8 and see how it can be grounded by linking its elements to a knowledge base. The representation can be visualized as a graph in Figure 2.9. It is easy to recognize the similarity with the predicate structure of Wikidata in Figure 2.3. We can simplify the representation by ignoring the event variables. Then, we link each participant to an entity in Wikidata and each edge between participants to a

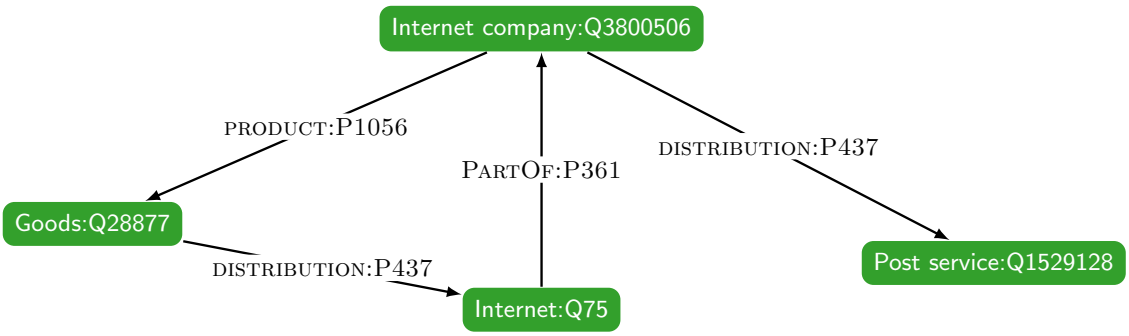

Figure 2.10: The logic-based representation from Figure 2.8 for the sentence "Online companies sell goods over the internet and they rely on the post service." visualized as a graph that uses entities and relation types from Wikidata.

binary Wikidata relation type. The result of this process would be a semantic representation in Figure 2.10, which is a grounded semantic graph. We call the process of constructing a grounded representation *grounded semantic parsing*.

We can rewrite the graph in Figure 2.10 as a logic formula. The Wikidata entities are individual constants and the Wikidata relations are binary predicates[10]:

$$\begin{aligned}&\text{product:P1056}(\text{Internet company:Q3800506}, \text{Goods:Q28877})\\ &\wedge \text{PartOf:P361}(\text{Internet company:Q3800506}, \text{Internet:Q75})\\ &\wedge \text{distribution:P437}(\text{Goods:Q28877}, \text{Internet:Q75})\\ &\wedge \text{distribution:P437}(\text{Internet company:Q3800506}, \text{Post service:Q1529128})\end{aligned} \tag{2.2}$$

To define formally, what Wikidata-based representations are possible, we write down the grammar of our semantic language using the BNF[11] notation (van Eijck and Unger, 2010):

$$\begin{aligned}\mathbf{v} &\rightarrow a \mid \ldots \mid z \mid \mathbf{v}'\\ \mathbf{c} &\rightarrow \text{Q1} \mid \ldots \mid \text{QN}\\ \mathbf{a} &\rightarrow \mathbf{v} \mid \mathbf{c}\\ \mathbf{P} &\rightarrow \text{P1} \mid \ldots \mid \text{PN}\\ \mathbf{F} &\rightarrow \mathbf{P}(\mathbf{a}, \mathbf{a}) \mid \neg\mathbf{F} \mid \mathbf{F} \wedge \mathbf{F} \mid \mathbf{F} \vee \mathbf{F} \mid \forall\mathbf{v}\mathbf{F} \mid \exists\mathbf{v}\mathbf{F}\end{aligned} \tag{2.3}$$

This definition combines the syntax of the first-order logic (foremost it allows quantification and logical operators to be applied to formulas) with individual constants that are entities from Wikidata and binary predicates that are Wikidata relations types. Recall the definition of Wikidata as a graph $W = (E, R, I)$ in Section 2.1. The grammar definition in Equation 2.3 allows to have infinitely many variables through the recursive definition of $\mathbf{v}$. The set of individual constants

[10] It is also possible to further reinterpret the phrase 'Online companies' as universal quantification: $\forall x.\text{instance of:P31}(x, \text{Internet company:Q3800506})$. We omit it here for simplicity.

[11] `https://en.wikipedia.org/wiki/Backus-Naur_form`

is limited to the range of entities in Wikidata E (Q1 | ... | QN, where we use QN for the largest numerical entity identifier in Wikidata), and the set of binary predicates is limited to the Wikidata relation types R (P1 | ... | PN with PN for the largest numerical relation type identifier in Wikidata). The following formulas are all allowed in our grounded semantic language:

$$\text{capital:P36}(\text{Germany:Q183}, \text{Berlin:Q64}) \tag{2.4a}$$

$$\exists x.\, \text{InstanceOf:P31}(x, \text{city:Q515}) \wedge \text{capital:P36}(\text{France:Q142}, x) \tag{2.4b}$$

$$\neg\text{capital:P36}(\text{Germany:Q183}, \text{Paris:Q90}) \tag{2.4c}$$

$$\forall x.\, \text{InstanceOf:P31}(x, \text{country:Q6256}) \rightarrow \exists z.\, \text{capital:P36}(x, z) \tag{2.4d}$$

In Equation 2.4d, we make use of the abbreviated notation for implication $\rightarrow$, which stands for $\neg(\mathbf{F} \wedge \neg\mathbf{F})$. For simplicity, we will use visual graphical notation for semantic representations throughout the rest of the thesis as in the example in Figure 2.10.

Logic-based semantic languages are expressive and powerful to cover various linguistic phenomena, as it was made evident through previous works in formal semantics on co-reference (Kamp and Reyle, 1996), presupposition (Sandt, 1992), negation (Richter and Sailer, 2006; Fancellu et al., 2017) and more (van Eijck and Unger, 2010). This large body of work offers exciting further directions and applications of grounded logic-based representations. We sketch out some of them in Section 6.3. In this work, we are using the grounded semantic language to encode the linking of the text to Wikidata. In that regard, our semantic representation is most similar to those of Reddy et al. (2014, 2016) and of Yih et al. (2015, 2016), Bao et al. (2016), who have been using Freebase in a similar grounding role.

Grounding in the knowledge base provides the formal interpretability for the semantic representation, which can be leveraged to compute the truth value of the whole expression. A logical formula is interpretable in a world model $\mathcal{M} = (D, I)$ that consists of an interpretation domain D and an interpretation function I. The interpretation function maps the semantic expressions to truth values and thus allows us to evaluate semantic expressions in our model of the world and reason about it. The interpretation is defined recursively for all valid semantic expressions.

We rely on the first-order logic definitions of interpretability for logic operators and variable assignments to the domain (van Eijck and Unger, 2010). The individual constants Q1 | ... | QN are directly referencing the entities of Wikidata E, which are a part of the domain D. The only component that we have to formally define interpretability for is the functional application form $\mathbf{P}(\mathbf{a}, \mathbf{a})$ in Equation 2.3, where $\mathbf{P}$ is a predicate (a Wikidata relation) and $\mathbf{a}$ is an argument (a variable or an individual constant). We use the Wikidata graph $W = (E, R, I)$ to define the truth values for these expressions in the world model $\mathcal{M}$:

$$\begin{aligned} \mathcal{M} \models P(a, a) \qquad \text{iff} \qquad & (g(a), g(a)) \in \text{IN}(P), \\ & \text{IN}(P) = \{\langle e_1, e_2\rangle \mid \langle e_1, r, e_2\rangle \in I, r = P\} \end{aligned} \tag{2.5}$$

where g is an assignment function that maps each individual constant to itself and a variable to some element of the domain D. IN is an interpretation function that maps the predicate P to a set of entity pairs $\langle e_1, r, e_2 \rangle$ in Wikidata for which this relation holds.

Formal interpretability is the essential property of the semantic representation for our applications. For instance, to answer a factual question like 'What is the capital of Germany?', we will translate it into a simple formal representation capital:P36(Germany:Q183, x). With the interpretability of the predicate capital:P36 defined as above, we are able to retrieve the set of entities for x from Wikidata that make this expression true and thus to find the answer to the question. In practice, the grounded semantic representation can be converted into a SPARQL query to retrieve the information from the knowledge base stored in the RDF graphical format.

To summarize, the resulting semantic language is expressive and has a defined formal interpretability for its statements. It is in contrast to the semantic representations discussed in Section 2.2.1, which are either limited in expressivity or lack the formal interpretability. How much of the language expressiveness is used in practice depends on the particular task and the data at hand. For instance, we have already shown that we limit our language to binary predicates to align it with Wikidata, although predicates of another arity would be also possible. Hence, the logic-based formal semantics offers a framework for different types of semantic representations rather than a single fixed semantic language.

In this thesis, we focus on the practical NLP applications, such as question answering, fact verification and argument comprehension, using Wikidata. Thereby, we tap only into the subset of the defined semantic language that is needed for the respective applications and the corresponding data. However, we conjecture that this representation could be a basis for more NLP solutions that involve external knowledge and logic-based reasoning. We return to this outlook in the conclusion in Section 6.3.

2.2.3 From Ungrounded to Grounded Semantic Parsing

In the previous sections, we have seen how an ungrounded logic-based representation from Figure 2.8 can be grounded in Wikidata to arrive at a grounded representation in Figure 2.10. In this section, we discuss how this process can be automated as an NLP pipeline. In particular, we detail how the text structure can be linked to a knowledge base step by step and what conceptual tasks are to be solved in the process.

We have been referring to the process of constructing any semantic representation automatically as semantic parsing. It is useful to distinguish two subtypes of it: ungrounded semantic parsing (more often called open-domain semantic parsing, as in 'no fixed domain or world model') and grounded semantic parsing. The ungrounded semantic parsing step often comes first in the pipeline followed

by the grounding step (Berant et al., 2013; Reddy et al., 2014, 2016). We have followed the same processing order with the manual semantic parsing example in the previous section: first an ungrounded representation in Figure 2.8 and then a grounded graph in Figure 2.10. We visualize how such a pipeline would look like in the diagram in Figure 2.11 using the process in Reddy et al. (2016) as the leading example.

Reddy et al. (2016) define a semantic parser that builds first-order logic-based representations from syntactic dependencies and then grounds them in a knowledge base. Such an approach couples the initial representation with the syntactic structure. The meaning representation is constructed compositionally, whereas the process is guided by the syntactic structure of the sentence. We demonstrate the syntax-to-logic conversion on the left of Figure 2.11. Montague (1974) was the first one to suggest that semantic representations can be compositionally constructed guided by syntax. To enable the automatic ungrounded semantic parsing, Reddy et al. (2016) define a set of rules that convert words and syntactic dependencies into logical formulas.

The main challenge associated with grounded semantic representations that use a fixed unambiguous vocabulary is the resolving of the semantic ambiguity of input natural language phrases. That is, the natural language phrases that have the same meaning, no matter how different they are, should be mapped to the same representation using the same set of vocabulary elements. Starting from the ungrounded semantic structure, the problem is reduced to matching ungrounded graph elements to entities and relations in the knowledge base (the bottom red line in the center of Figure 2.11). Reddy et al. (2016) use a relation disambiguation and an entity linking model to convert the ungrounded structured semantic representation to a grounded one. Reddy et al. (2014) construct the same pipeline using a CCG syntactic parser (Clark and Curran, 2007) and Berant et al. (2013) use the same process but rely on Lambda Dependency-Based Compositional Semantics for the ungrounded logical forms. Entity linking and relations extraction are thus at the center of grounded semantic parsing and are the core components for linking parts of text to elements in the knowledge base. We discuss methods for entity linking and relation extraction as stand-alone tasks and as a foundation for grounded semantic parsing in Chapter 3.

The Reddy et al. (2016)'s approach focuses on grounding entities and relations, whereas the structure of the representation is derived from syntax. This view is tightly related to the compositional construction of logic-based representations following the syntactic structure as proposed by Montague (1974). However, a such pipeline may lead to error propagation from syntactic parsing. Moreover, structure changes to the representation may be needed at the grounding stage (e.g., simplifying the structure by merging two edges).

More recent approaches of Yih et al. (2015), Bao et al. (2016) and Peng et al. (2017) have attempted to generate a grounded semantic representation directly from the input text using only an entity linker as a dependency. An entity linker recognizes

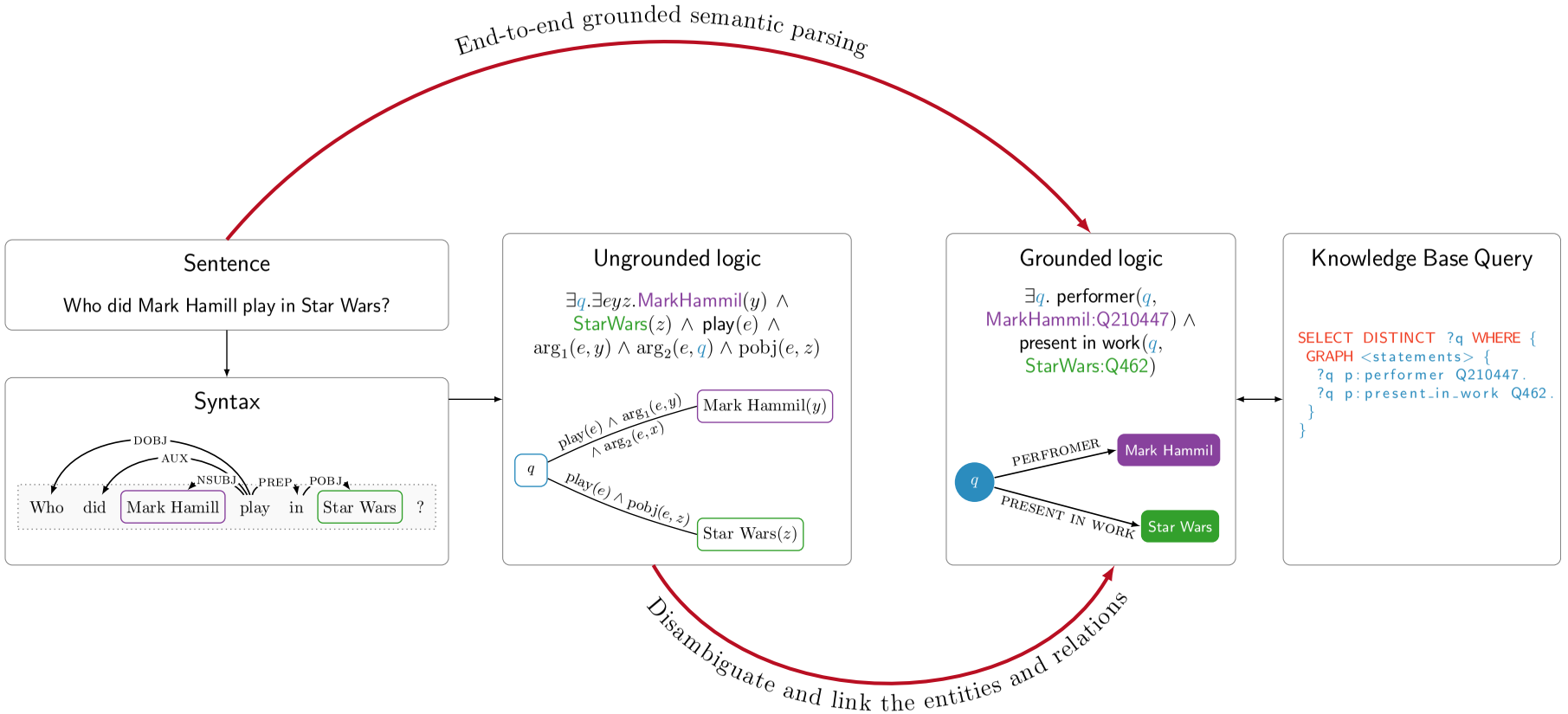

Figure 2.11: From extracting the ungrounded structure of a sentence to grounding it and converting it into a knowledge base query. The black links show conversion steps, the red links show the grounding: the relation disambiguation and entity linking step or a direct end-to-end grounded parsing.

knowledge base entity mentions in the text and its output can be used to limit the space of possible semantic representations: only the recognized entities and compatible relations types are allowed. Such approaches make a construction of a grounded semantic representation possible without an intermediate syntactic parsing step.

In this thesis, we describe a semantic parsing system for knowledge base question answering and other tasks that follows the approach of Yih et al. (2015), which does not rely on syntactic parsing to construct structured semantic representations. Our approach constructs semantic graph representations directly from the input without the intermediate discrete steps of syntactic parsing, ungrounded representation and linking. All those steps are instead incorporated into a single end-to-end architecture (the upper red line in the center of Figure 2.11). We present the advantages of such a system in Chapter 4. We view the grounded semantic representations as directed graphs (shown in Figure 2.10), which are generated by the semantic language defined in Section 2.2.2. To directly produce such semantic graphs from the input text, we rely on GNNs, which we discuss in the next section.

Semantic parsers and the data that they are learned from define the space of semantic representations that can be produced in practice. For instance, semantic graphs produced by our semantic parser in Chapter 4 do not include all aspects of the first-order logic language we have defined, such as negation, as those rarely occur in the question answering data that we use for training.

The data annotated with semantic representations is limited. As the result, there has been rising interest in learning semantic parsers from indirect supervision. The examples include unsupervised approaches that leverage distributional similarity by recursive clustering (Titov and Klementiev, 2011), semi-supervised approaches that learn from dialog context (Artzi and Zettlemoyer, 2011), grounded approaches that learn from annotated question-answer pairs (Liang et al., 2011; Berant et al., 2013) or virtual physical environments (Artzi and Zettlemoyer, 2013). We turn to weak supervision in this work to train our grounded semantic parser.

2.3 Graph Neural Networks

Graph Neural Networks (GNNs) generalize and extend the machine learning and neural approaches that have been developed for graphs and provide a direct solution for processing structured graphical information with neural methods (Battaglia et al., 2018). The models in this family of neural networks have been studied and developed for the last decade and the NLP community has started adopting GNNs actively in the last three years (Marcheggiani and Titov, 2017; Schlichtkrull et al., 2018; Sun et al., 2018).

In this section, we present the motivation and the inductive biases of the GNNs.

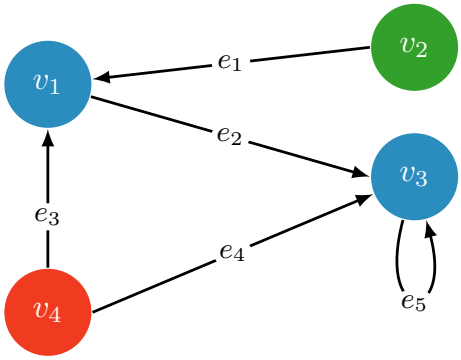

Figure 2.12: A basic graph with three node types, four nodes and five edges.

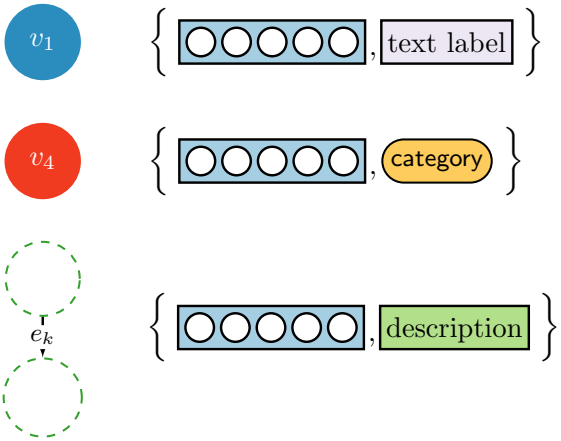

Figure 2.13: Graph nodes and edges with different attributes: a label, a category or a description.

We introduce two variants of graph networks that have been the basis of the most approaches in the NLP research and were concretely applied to knowledge bases: Graph Convolutional Network (GCN) and Gated Graph Neural Network (GGNN). We adopt the latter variant in our work. We give the reasons for adopting GGNN and provide the mathematical formulation for this type of models in this section. We also touch upon the crucial implementation strategies for GNNs, such as storing the sparse graph adjacency matrix efficiently.

2.3.1 Graphs

Before we start with the description of graph networks, we define a type of graph that can be processed and its basic properties. We regard graphs as a set of nodes $\mathcal{V}$ and a set of edges $\mathcal{E}$. Each edge $e \in \mathcal{E}$ links two nodes $v \in \mathcal{V}$ (always a binary edge) and has a direction. It might connect a node to itself, thus the same node is both the start and the end of the edge, i.e. a loop. Each node and edge might have a type (a category) and can have further information attached to them, such as a label or a textual description. The graphs are allowed to contain cycles.

Figure 2.12 shows an example of a graph that a GNN is able to process. It has

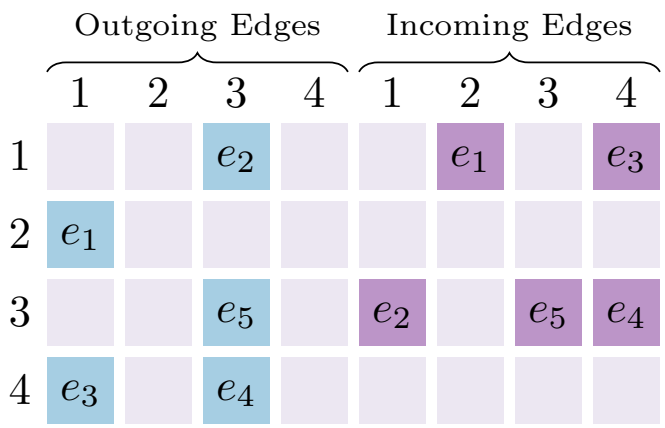

Figure 2.14: Adjacency matrix encoding of the graph structure in Figure 2.12.

four nodes of three different types and five edges, including a single self-loop. Each edge and node have a unique index and all edges have a direction. In Figure 2.13, we give examples of attribute sets that an element of a graph (a node or an edge) can have: it might be an associated dense vector that encodes multiple properties, a text label, a category type, a free text description or a combination of those. The attributes associated with the nodes and edges serve as the input to the GNN that then transforms them following the internal graph structure into vector encodings for the nodes and the edges and into a summary encoding for the whole graph.

We can see from the above definition that the semantic graph representations grounded in the Wikidata knowledge base, as described in the previous section, would fit the given definition. Wikidata itself is a large graph composed of entities and binary relations. Thus, a subgraph of Wikidata could be processed with a GNN. The nodes would be defined from the set of Wikidata entities and have global IDs and textual labels attached as their attributes. The edges would have a category that is one of the relation types from the Wikidata schema and a textual description.

We need to encode the graph attributes and its structure in a way that can be processed by a neural network. The node and edge attributes can be embedded and stored as multi-dimensional vectors. We will encode the graph structure with an adjacency matrix (see an example in Figure 2.14). An adjacency matrix is a common way to store the connections between elements in a sparse integer matrix $\mathbf{A} \in \mathbb{Z}^{|\mathcal{V}|\times 2|\mathcal{V}|}$, where the integer index value is used to differentiate node types and 0 stands for no edge (Battaglia et al., 2018). It contains a row for every node in the graph and twice as many columns (for outgoing and incoming edges). For each node, we store its outgoing and incoming edges in the node's row. Thus, to store a binary edge $\langle v_2, e_1, v_1 \rangle$ in the adjacency matrix, we find the second row, which corresponds to the node v_2 and put the type of the edge e_1 into the first column, which encodes that it is an outgoing edge towards v_1. We also have to store the same information in the row for the node v_1 as an incoming edge.

We take the first row of the matrix and put e_1 into the sixth column (the second column of the 'incoming edges' part of the matrix), which corresponds to v_2.

We use the described adjacency matrix as a basis for storing the graph structure in a numerical form. However, the default adjacency matrix is sparse, since most of the graphs are not densely connected. The adjacency matrix that is mostly zeros becomes inefficient to store and to process in the practical setting. The modern deep learning frameworks, such as TensorFlow[12] and PyTorch[13], support sparse matrix objects with better memory efficiency, but the neural network engine still requires a dense matrix as an input to process the gradients correctly. This becomes especially relevant in the knowledge base semantic parsing context, where the goal might be to evaluate hundreds of semantic graph hypotheses per single input. In the context of the experiments described in the later chapters, we will encounter hundreds of graphs per single training instance in the process of searching through Wikidata subgraphs. Hence, the memory efficiency requirement is crucial for the practical implementation. We modify the adjacency matrix to better fit the needs of the GNN architecture and to balance the memory efficiency in the next sections.

2.3.2 Relational Graph Convolutional Networks

To motivate our choice of GGNNs, which is a type of recurrent architecture in the next section, we first describe another prevalent GNN architecture that is based on convolutions.

Relational Graph Convolutional Networks (R-GCNs) were developed in the network family of GCNs (Kipf and Welling, 2017). They are used for processing knowledge base information and specifically for computing structure-induced entity and relation embeddings. The original GCNs were designed to handle directed graphs with different relation types, but could only incorporate a small set of relations at once, since each relation required a separate matrix to learn. The main idea of the graph convolutional processing was to apply convolutions for each node on the surrounding nodes and relations and to use the aggregated information from the convolution operation to update the vector encoding of the node in question. Thus, it transforms the idea of the traditional convolutional networks, which apply the convolution operation based on spatial proximity (Goodfellow et al., 2016, Chapter 9), into convolutions on graphs. Marcheggiani and Titov (2017) employed GCNs for natural language processing for the first time to encode syntactic graphs.

GCNs can be understood as a special case of a differential message-passing framework (Gilmer et al., 2017). Let us process each graph node i with l layers that accumulate the information about the neighbors and let us denote the output of each processing step with the hidden vector h_i^l. The update equation for h_i^l in the

[12] `https://www.tensorflow.org/api_docs/python/tf/sparse/SparseTensor`
[13] `https://pytorch.org/docs/stable/sparse.html`

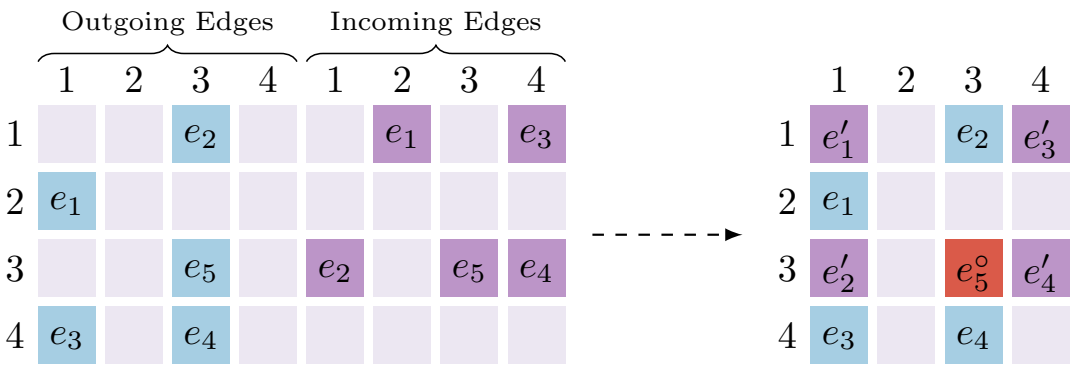

Figure 2.15: Adjacency matrix encoding of the graph structure transformed into a more compact representation using different identifiers for the inverse of the relations.

message-passing framework would then be defined as:

$$h_i^{l+1} = \sigma\Big(\sum_{m \in M_i} g_m(h_i^l, h_j^l)\Big) \tag{2.6}$$

where M denotes the set of incoming messages from the neighboring nodes in the graph that are transformed with g_m. The information on the neighboring nodes is stored in the adjacency matrix $\mathbf{A} \in \mathbb{Z}^{|\mathcal{V}| \times 2|\mathcal{V}|}$. To make the adjacency matrix more compact, inverse relations are given a special identifier, which results in $\mathbf{A} \in \mathbb{Z}^{|\mathcal{V}| \times |\mathcal{V}|}$ that is twice that small (Figure 2.15).

R-GCNs is an extension of graph convolutional networks that can incorporate thousands of relation types through the decomposition of relation matrices into a fixed set of learnable components. R-GCNs were tested on modeling links in a knowledge base (Schlichtkrull et al., 2018). The resulting networks learn structure-induced embeddings for knowledge base entities and use them to predict further links for automatic knowledge base completion. Schlichtkrull et al. (2018) have taken on the problem of modeling the hundreds of knowledge base relations at once and predicting the missing connections in the knowledge base. Conceptually, their model creates a vector encoding for the node that is learned to represent the surrounding relations and nodes. Then, the node vector encodings are iteratively updated using the encodings of the surrounding nodes from the previous iteration, which is the underlying idea of message-passing networks and of the graph convolutional networks. By computing iterative updates, the information is propagated from one node to another and each node contains information about its neighbors after converging.

The challenge is to adopt the message passing mechanism in Equation 2.6 to the knowledge base and to support hundreds of different relation types. Schlichtkrull et al. (2018) define the following form for the message-passing equation:

$$h_i^{l+1} = \sigma\Big(\sum_{r \in R} \sum_{j \in N} \frac{1}{c_{i,r}} W_r^l h_j^l + W_0^l h_i^l)\Big) \tag{2.7}$$

where N is the set of neighbors for the i-node and R is the set of relation types in the knowledge base. The above equation requires a separate transformation matrix W_r to be learned for each relation type in the knowledge base. With hundreds of relation types, the number of parameters to be learned explodes. Schlichtkrull et al. (2018) overcome this by redefining W_r as a linear combination of a fixed set of basis matrices:

$$W_r^l = \sum_{b=1}^{B} a_{rb}^l V_b^l \tag{2.8}$$

where V_b is a basis component. This reduces the number of parameters needed to be learned to a fixed set of size B. Schlichtkrull et al. (2018) use the vectors learned for entity nodes and relation matrices to predict missing links for nodes that are similar as per their vector encodings.

For the purpose of knowledge base semantic parsing, the relational graph convolutional networks have a major limitation. Schlichtkrull et al. (2018) decompose the relation matrix into a weighted combination of basis transformations and learn different weights for each relation type. This increases the model capacity to learn a new embedding for each of the hundreds of relation types, but it does not incorporate the relation descriptions or other lexical information that is crucial to overcome the lexical gap in knowledge base semantic parsing and question answering (Jain, 2016). In the next section, we turn to GGNNs, which exhibit most of the same properties but offer more flexibility in encoding the structure of the graph and of its individual elements.

2.3.3 Gated Graph Neural Networks

In this work, we adopt GGNNs for the processing of semantic graphs, for learning vector representations of entities, relations and of the whole graphs and ultimately for our semantic parsing and question answering architectures. GGNNs, similarly to GCNs, process graphs by iteratively updating representations of graph nodes based on the neighboring nodes and relations (Li et al., 2016) and can be understood in terms of the message-passing framework (Equation 2.6). The starting point for their development is the GNN formulation of Scarselli et al. (2009). They describe a propagation model that is akin to the recurrent neural networks (Goodfellow et al., 2016, Chapter 10) rather than to convolutional architectures. The GGNNs maintain a separate hidden state that is used to compute the output vectors for the nodes and for the overall graph and include gating mechanisms to control the passing of information.

Developed in parallel to the GCNs, GGNNs have an equivalent expressivity. They are able to incorporate the same information from the graph structure into the vector representations, but offer more flexibility than GCNs as to how this information is transferred and stored between the individual vector representations due to the employed gating mechanisms. Li et al. (2016) give a formulation of

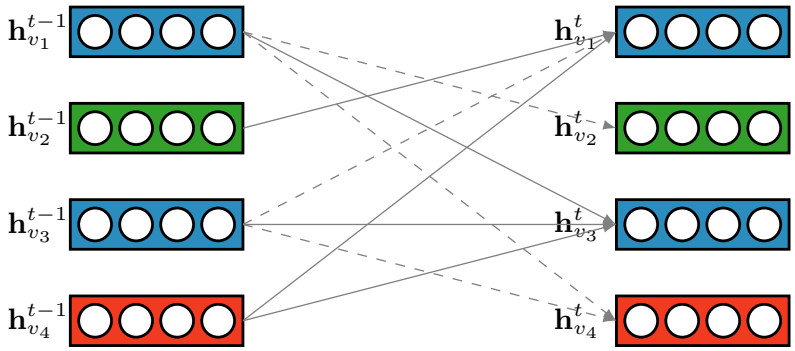

Figure 2.16: Unrolled GGNN recurrence for one time step for the graph in Figure 2.12. Solid lines show updates along the direction of the relation in the graph and the dashed lines in the opposite direction.

GGNNs for graphs with labeled nodes and typed directed edges. We extend Li et al. (2016)'s formulation to include labeled edges and apply it to grounded semantic graphs in Chapter 4. To the best of our knowledge, we are the first to apply GGNNs to semantic parsing and knowledge base question answering. In this section, we present the Li et al. (2016)'s definition of GGNNs to provide the necessary background for our work. Similarly to R-GCNs, it only supports typed edges and we turn to our extension for knowledge base relation labels and the empirical evaluation in Chapter 4.

Consider again the basic graph in Figure 2.12. Li et al. (2016) describe a network architecture that can process graphs of arbitrary size, of any number of nodes and edges and that takes the directionality of edges $e_1, \ldots, e_5$ and the node types $v_1, \ldots, v_4$ into account. It does not incorporate edge types and can only support node labels if those are encoded as node types/categories. The hidden vector for each node $\mathbf{h}_v^{(1)}$ is initialized from the node properties (Figure 2.13).

Propagation Model GGNNs are a type of recurrent neural networks and hence, Li et al. (2016) define a propagation model from time step $t-1$ to t. The recurrence is unrolled for a fixed number of steps T and the gating mechanism for the hidden layers is adopted from Gated Recurrent Units (Cho et al., 2014). Formally, the propagation model for GGNNs is defined as follows:

$$\mathbf{a}_v^{(t)} = \mathbf{A}_{v:}^{\top}\left[\mathbf{h}_1^{(t-1)\top} \ldots \mathbf{h}_{|\mathcal{V}|}^{(t-1)\top}\right] \tag{2.9}$$

$$\mathbf{z}_v^t = \sigma\left(\mathbf{W}^z \mathbf{a}_v^{(t)} + \mathbf{U}^z \mathbf{h}_v^{(t-1)} + \mathbf{b}^z\right) \tag{2.10}$$

$$\mathbf{r}_v^t = \sigma\left(\mathbf{W}^r \mathbf{a}_v^{(t)} + \mathbf{U}^r \mathbf{h}_v^{(t-1)} + \mathbf{b}^r\right) \tag{2.11}$$

$$\widetilde{\mathbf{h}_v^{(t)}} = \tanh\left(\mathbf{W}\mathbf{a}_v^{(t)} + \mathbf{U}\left(\mathbf{r}_v^t \odot \mathbf{h}_v^{(t-1)}\right) + \mathbf{b}\right) \tag{2.12}$$

$$\mathbf{h}_v^{(t)} = (1 - \mathbf{z}_v^t) \odot \mathbf{h}_v^{t-1} + \mathbf{z}_v^t \odot \widetilde{\mathbf{h}_v^{(t)}}, \tag{2.13}$$

where σ is the logistic sigmoid function and $\odot$ is the element-wise multiplication. Thereby, one recurrence step passes the messages from and to the neighboring nodes (compare the Equation 2.12–2.13 with the message passing Equation 2.6).

The adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times 2|\mathcal{V}|}$ stores the structure of the graph: a row of the matrix $\mathbf{A}_{v:}$ records the edges between the node v and the other nodes in the graph. The activations $\mathbf{a}_v^{(t)}$ for each node are computed based on the neighboring node hidden vectors $\mathbf{h}_v^{(t-1)}$ (Equation 2.9). The $\mathbf{z}_v^t$ in Equation 2.10 and $\mathbf{r}_v^t$ in Equation 2.11 are update and reset gates that incorporate information from the previous step to update nodes' hidden states at each iteration.

The above definition uses the matrix $\mathbf{W}$ in Equation 2.12 to transform the activations from the neighboring nodes. To support multiple relation types, a single $\mathbf{W}$ can be learned per relation type. Similarly to GCNs, this solution does not scale to thousands of knowledge base relations. It is possible to apply the same basis matrices trick as used by R-GCNs here. Yet, as we have already noted in the previous section, such a solution still does not enable the incorporation of relation type properties such as a relation description. To use the relation description in the graph structure, one would need to introduce relation type hidden vectors to encode this information. In Section 4.4.2, we propose two solutions to extend GGNNs with relation vectors and verify them empirically on the knowledge base question answering task.

Graph-level Output Vector The recurrence is unrolled for a fixed number of steps T in the experiments (Figure 2.16). From the hidden vectors for each node, we can compute node output vectors. To produce a graph-level output vector, Li et al. (2016) suggest to take the sum of hidden node vectors at the last time step $t = T$ and transform it with a fully-connected layer and the ReLU non-linearity: $\mathbf{v}_g = \text{ReLU}\big(\mathbf{W}_\mathbf{g}(\sum_{v \in \mathcal{V}} \mathbf{h}_v^{(t)}) + \mathbf{b}_\mathbf{g}\big)$.

To store the structure of the graph, Li et al. (2016) use the unmodified adjacency matrix as in Figure 2.14, which is inefficient for large graphs. In this thesis, we solve this issue by decomposing the adjacency matrix $\mathbf{A} \in \mathbb{Z}^{|\mathcal{V}| \times 2|\mathcal{V}|}$ into two components (see Figure 2.17): the one that stores the node ids of the neighboring nodes $A_v \in \mathbb{Z}^{|\mathcal{V}| \times N_v}$ and the one that stores the corresponding relation type ids $A_r \in \mathbb{Z}^{|\mathcal{V}| \times N_v}$, where N_v is the maximum number of the neighbors for a single node. The result are the two matrices that together are smaller than the original adjacency matrix as long as the graph is not fully connected, that is $N_v < |\mathcal{V}|$. Moreover, on knowledge base data where $N_v << |\mathcal{V}|$ this solution is more memory efficient than the adjacency matrix transformation in Figure 2.15 as used for R-GCNs. The matrix in Figure 2.15 is $\mathbf{A} \in \mathbb{Z}^{|\mathcal{V}| \times |\mathcal{V}|}$ and the two matrices in Figure 2.17 have a combined size of $|\mathcal{V}| \times 2N_v$, so each node has to be connected to less than a half of other nodes in order for our solution to be more memory efficient. Consider the example in Figure 2.3, there are five nodes in total $|\mathcal{V}| = 5$ and each one is connected to two other nodes at maximum $N_v = 2$. The resulting

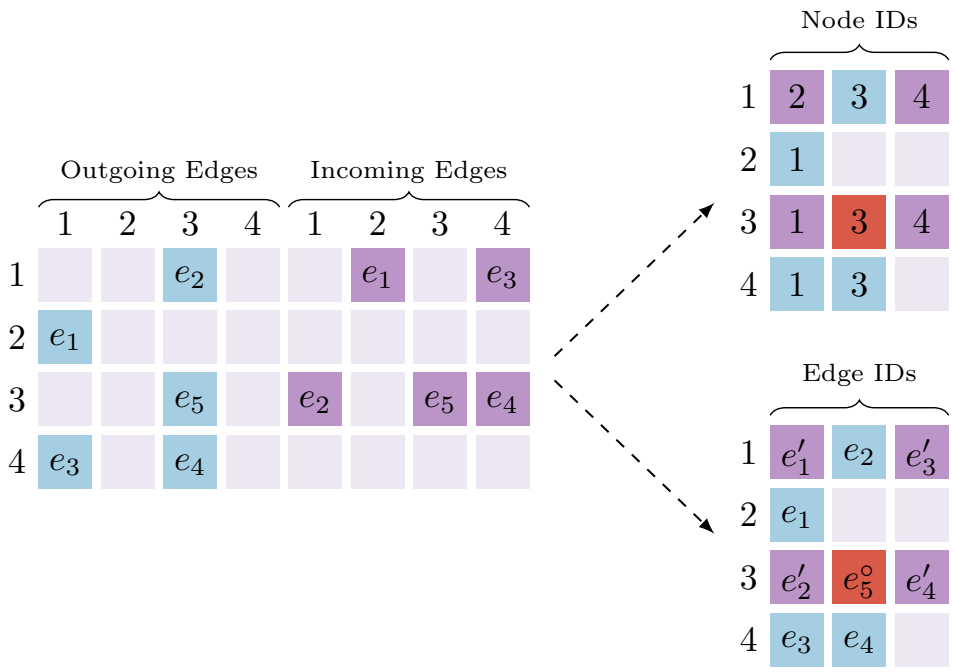

Figure 2.17: Adjacency matrix encoding of the graph structure decomposed into two smaller matrices. The new matrices store node and edge IDs separately. The decomposition results in a smaller overall memory footprint when the graph size grows.

accumulated size of our decomposition is $|\mathcal{V}| \times 2N_v = 20$, whereas the R-GCNs option in Section 2.3.2 amounts to $|\mathcal{V}| \times |\mathcal{V}| = 25$. We use the decomposed solution in the practical implementation of GGNNs for the experiments in this thesis.

We are the first to apply GGNNs to grounded semantic graph modeling and semantic parsing. The results of our first empirical experiments with GGNNs were published in Sorokin and Gurevych (2018a). These experiments are described in detail in Chapter 4. Concurrently, Sun et al. (2018) have explored the combination of knowledge base information with text to answer open-domain questions. They link knowledge base statements to the document paragraphs and apply GGNNs to score each entity in the linked statements as a potential answer in a document question answering setup (Miller et al., 2016). They do not explore grounded semantic parsing with GGNNs.

2.4 Chapter Summary

In this chapter, we have introduced the concepts, resources and machine learning frameworks that are at the core of our contributions. Most importantly, we have first focused on the Wikidata knowledge base and its comparison with Freebase. The goal of this work is linking the text to a knowledge base and using the model of the world encoded in the knowledge base for practical NLP applications. Wikidata

is a community-managed rapidly developed knowledge base that is not restricted to a single domain and has a well-defined schema of entities and binary relation types. These properties make it an attractive resource for NLP.

Linking the text to the knowledge base is essentially interpreting it with respect to the knowledge base's model of the world. We formalize this process as constructing an explicit semantic representation that is grounded in (or linked to) the knowledge base. We have introduced a set of different semantic representations that have been used in NLP in the past and have motivated the use of logic-based semantic language. We have defined the logic-based semantic language grounded in Wikidata, which we can use to encode a complete linked meaning representation of the input text. The resulting grounded representation is a semantic graph, which we use throughout this thesis as the main building block.

We have introduced the process of constructing a logic-based representation and then grounding it in the knowledge base in Section 2.2.3. The process is structured as a step-by-step pipeline that converts the input into ungrounded semantic representation and then employs the linking methods to ground it. In the next Chapter 3, we present novel architectures for linking individual elements to the knowledge base. The multi-step grounding pipeline may suffer from error propagation. Hence, we have motivated the approach that produces grounded semantic graphs directly from the input sentence surpassing the syntactic parsing step. We build upon previous work in this direction and present our semantic parser in Chapter 4.

Our semantic parser relies on the powerful framework of GNNs, which we have introduced in the final section of this chapter. Graph Neural Networks have the power to process arbitrary graph structures, but there are limitations with respect to the relation type information that can be incorporated. We have reviewed the R-GCNs architecture, which was developed for link prediction in knowledge bases, and the GGNNs recurrent model. As one of the main contributions of this dissertation, we adapt GGNNs for Wikidata in Chapter 4.

Chapter 3

Linking Text to a Knowledge Base

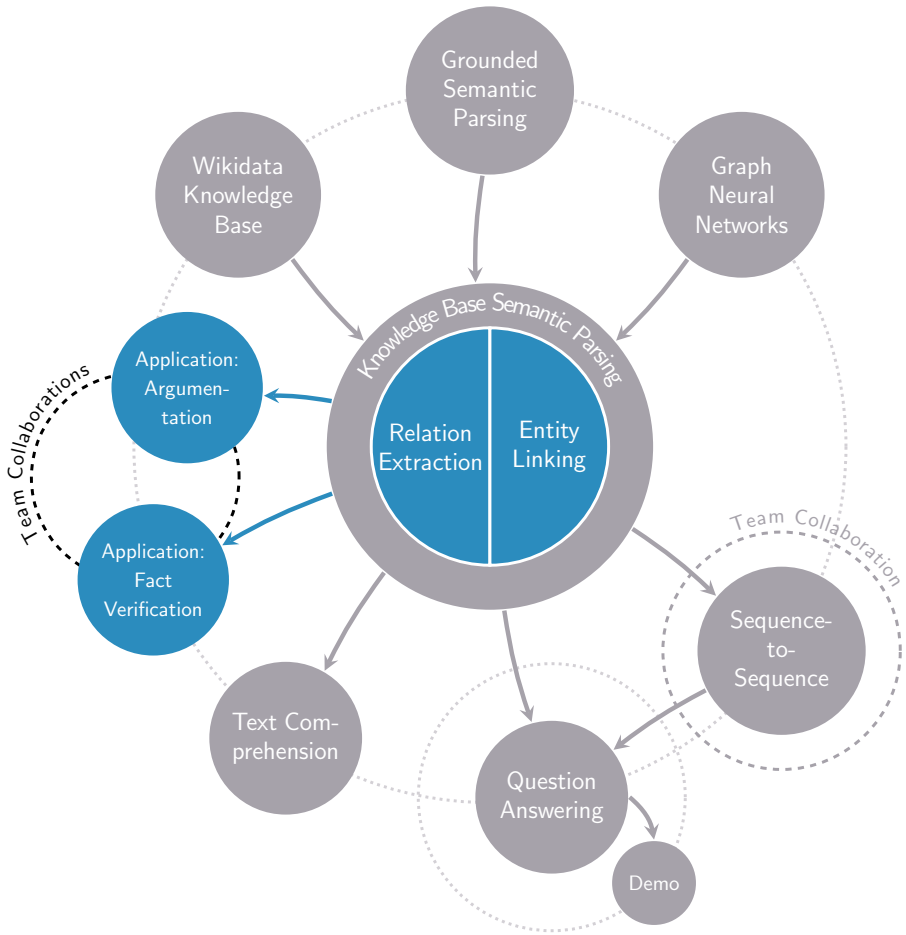

Figure 3.1: Chapter 3 in the thesis diagram. This chapter discusses new methods for linking relation and entity mentions in the text to the knowledge base (the blue semicircles at the center of the diagram). The entity linker is extrinsically evaluated on two NLP applications: argument reasoning comprehension and fact verification (the blue circles on the left).

A knowledge base is encoded as a graph, where the vertices are world entities and edges are semantic relations. This chapter describes the two main building blocks for linking text to the knowledge base: a relation extraction system and an entity linker. The relation extraction task is to classify the occurrences of semantic relations in texts. The entity linker focuses on the identification and disambiguation of entity mentions. These two components form the basis of a grounded semantic parsing pipeline, so that the identified entity mentions

and relations are used to construct a full semantic graph representation of an utterance (relation extraction and entity linking form the step from ungrounded representation to grounded in Figure 2.11). This is the path that we follow in the next chapter in the question answering setting. Altogether, this chapter addresses the first research question that we have formulated in the introduction:

RQ1 What are the challenges for linking a text to an external knowledge base and how can the linking methods be improved?

We show that existing methods for relation extraction and entity linking lack a comprehensive modeling of the surrounding context. We present two novel methods in this chapter (one for each of these tasks): a relation extraction model that considers multiple relations in the same context together and an entity linking architecture that aggregates contextual information from the input sentence and from the knowledge base in a single architecture. Our methods advance the state of the art on the respective benchmarks and enable more efficient downstream applications. We developed our methods in the broader context of the semantic parsing pipeline and with the requirements of downstream tasks in mind. Our main application task throughout the thesis is question answering.

Entity linking and relation extraction may be also used independently to extract meaningful information from texts for further processing and analysis by data experts or in a form of knowledge-informed features in NLP applications. In the final section of the chapter, we discuss concrete practical applications of the introduced methods. We apply our novel entity linking method with context aggregation on two NLP tasks: argument reasoning comprehension and automatic fact verification. First, we combine the knowledge base information with the frame semantic annotations to add the world knowledge context to a state-of-the-art argument reasoning system. Second, we use entity linking to find the supporting information for the extraction of evidence in a fact verification pipeline. It has been shown in the previous analyses that such application tasks require external world knowledge. In this chapter, we present positive empirical results for these tasks with our entity linking system. To this extent, the application sections of this chapter offer first evidence towards the third research question of this thesis:

RQ3 Can the knowledge base linking methods and the grounded semantic parser generalize to new data and be embedded into natural language understanding applications?

The chapter is organized into three large sections. The first section 3.1 discusses the relation extraction task and our context-aware architecture. The second section 3.2 introduces entity linking for question answering and our approach designed to work for noisy data and across multiple entity categories. In the last section of this chapter (Section 3.3), we consider two downstream applications that require external world knowledge and provide empirical evidence that they can be improved with linking the input to Wikidata.

3.1 Relation Extraction

Relation extraction is a fundamental semantic task that enables a wide range of applications from question answering (Xu et al., 2016) to fact checking (Vlachos and Riedel, 2014). In this section, we describe relation extraction experiments and examine the special case of the occurrence of multiple relations per sentence and challenges that it poses. Multiple relations per sentence occur frequently in the fact checking and question answering data. However, this situation poses a considerable challenge for the current NLP systems, as they usually operate under an assumption that one sentence expresses only a single relation (Riedel et al., 2010; Zeng et al., 2015). Thus, they are unable to correctly extract multiple relations from the same sentence. For instance, we show in Section 4.1 that questions that involve more than one semantic relation pose a particular problem to the current approaches. One entity may participate in multiple relations in the same sentence or several relations between different entities may overlap in the same sentence. We detail this task requirement and discuss the downstream applications in the next section.

Methodologically, we demonstrate that for sentence-level relation extraction it is beneficial to consider other relations in the sentential context while predicting the target relation. That is, although multiple relations in the same sentence pose processing challenges, we show a way to use this information to improve the overall performance. Our architecture uses a neural network encoder to jointly learn vector representations for all relations in a single sentence. We combine the relation vectors with an aggregation mechanism that assigns an individual weight to each vector to make the final prediction.

To evaluate our approach, we contribute a new distantly supervised relation extraction dataset. We use the Wikidata knowledge base to construct a dataset of multiple relations per sentence. The new dataset enables an empirical comparison of the proposed relation extraction model with the existing approaches, which we take as a baseline. Our method results in an average error reduction of 24 % on a held-out set of relations in the constructed dataset.

3.1.1 Task Description and Motivation

The main goal of relation extraction is to determine the type of relation between two target entities that appear together in a text. We consider the relation extraction task on the sentence level: to each occurrence of the target entity pair $\langle e_1, e_2 \rangle$ in some sentence s one has to assign a relation type r, which is expressed in the given sentence (Hoffmann et al., 2011). The possible relation type r is restricted to a fixed set R. The goal is thereby to extract a relation that exists between the entities based on the content of the input text (see Figure 3.2 for an example) and not to recover a relation that generally holds between the two entities .

It is useful to note that this task definition is different from the wide-spread

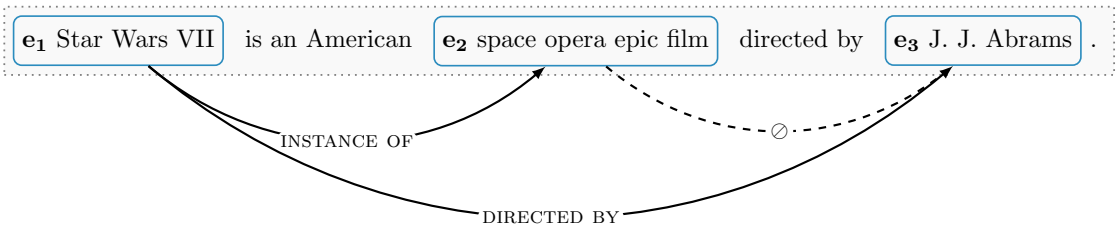

Figure 3.2: An example input sentence and output for extracting multiple relations from the same sentence. Multiple relations overlap in the single sentence in this example and it is thus not trivial to determine the best encoding of the context to correctly predict a relation between each of the three pairs of entities.

relation extraction task definition in Riedel et al. (2010) and the follow-up work. Riedel et al. (2010) aim to recover a relation between two entities $\langle e_1, e_2 \rangle$ based on an aggregated evidences from all co-occurrences of the $\langle e_1, e_2 \rangle$ in a corpus of texts. In that setting, it is not required or important that a particular sentence expresses the extracted relation. Overall, recovering a relation between the two entities based on the collected evidence as in Mintz et al. (2009) and in Riedel et al. (2010) is tied to the knowledge base completion task and information extraction in general. Their goal thereby is to collect information about relations, but not to analyze or to interpret individual sentences. We focus instead on the relation extraction as one of the means to disambiguate the input meaning.

Our perspective is tied to the semantic parsing view of representing the meaning of a sentence through entities and relations between them (see the discussion in Section 2.2). Our goal in this section is the classification and extraction of individual relations so that it is possible to link a particular piece of text to a knowledge base. We determine the relation type based on a single sentence input and thus also consider this to be a subtask of semantic parsing rather than a standalone task. Among the previous work, Hoffmann et al. (2011) and Surdeanu et al. (2012) assign multiple relation types to each entity pair, such that the predictions are tied to particular occurrences of the entity pair. That is a similar setup to our use case.

It is through the link between the input sentence and the extracted relation that is expressed in it that the defined relation extraction task setup can be useful for question answering, automatic fact verification and other tasks. For instance, for question answering, we can extract relations from the question by paring the question word with the entity mentions (see Figure 3.3). We can find an answer by consulting the knowledge base and finding an entity that is related to the entities in the question in the same way. Similarly, in the context of fact verification, we can extract the relation that is asserted in the given input sentence between two entities and compare it with the relations between the same two entities in the existing knowledge base or with other entities that are connected with the same relation to one of the target entities (see Figure 3.4). If some of the information

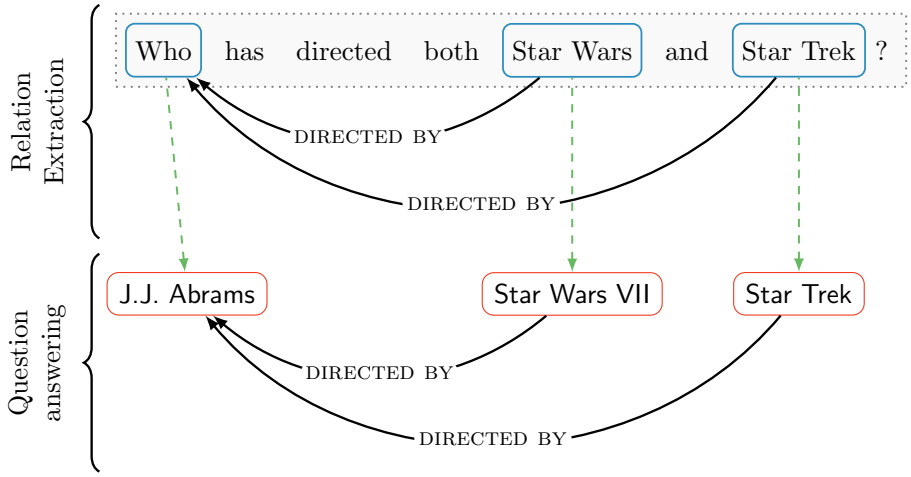

Figure 3.3: Relation extraction for question answering. This figure illustrates the effect of the two extracted relation instances on the downstream question answering task. The upper part shows two relations between the entities and the question word in the sentence. In the lower part, this information is compared against the reference in the knowledge base to find an answer entity that is connected to the entities with the same relations.

does not match up, we can conclude that either the new information is incorrect or the knowledge base (we come back to the fact verification task in section 3.3.2 and to the knowledge base question answering task in chapter 4).

Figure 3.3 also shows a further example input where multiple relations overlap in the same sentence. Such cases are the particular focus of our contribution in this section. More than 30 % of questions in the common question answering dataset, WebQuestions (Berant et al., 2013), require multiple relations to find the correct answer. And in the most recent fact verification dataset, FEVER (Thorne et al., 2018), 17 % of the claims require a composition of multiple pieces of evidence, which means that the claim itself contains more than one relation instance. These observations from the potential downstream applications allow us to formulate two requirements for our relation extraction setup: (i) the relation extraction architecture should, for a given sentence and a pair of entity mentions, extract the relation that is expressed in this sentence or no relation if the input sentence does not describe a relation between the entities; (ii) the relation extraction architecture should be robust for contexts that describe multiple relation instances.

We call a triple $\langle e_1, r, e_2 \rangle$ a *relation instance* and we refer to the relation of the target entity pair as a *target relation*. Figure 3.2 shows an example input sentence with three entities: e_1, e_2, e_3. We take the empty relation $\oslash$ to be part of the relation set R. Consequently, we consider every pair of entities in the same sentence to form a relation instance: $\langle e_1, r_{e_1,e_2}, e_2 \rangle, \langle e_1, r_{e_1,e_3}, e_3 \rangle, \langle e_1, r_{e_2,e_3}, e_3 \rangle$, whereas $r = \oslash$ would denote an empty or no relation: $\langle e_1, \oslash, e_3 \rangle$. There is an empty relation between 'J.J. Abrams' e_3 and 'space opera epic film' e_2 in the example in Figure 3.2, as there is no relation between them expressed in this sentence. At the same time, the relations

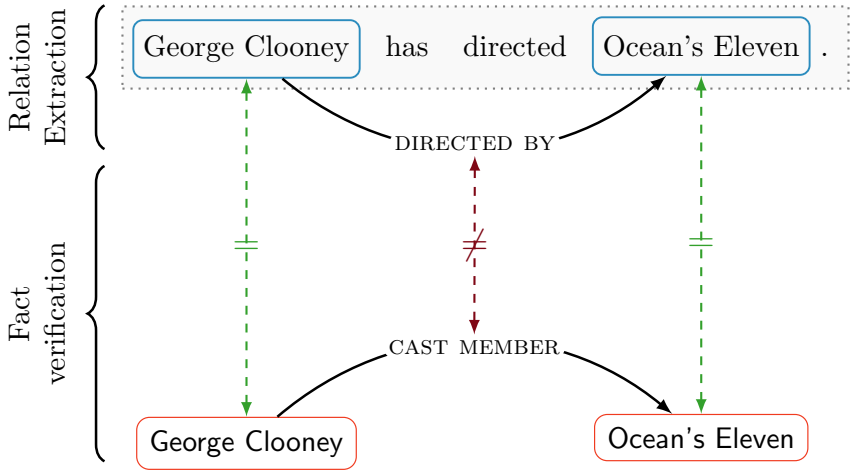

Figure 3.4: Relation extraction for fact checking. The upper part shows a relation between two entities that is then compared against the reference knowledge in a knowledge base (the lower part).

between the e_1 'Star Wars VII' and the e_2 'space opera epic film' (INSTANCE OF) and between e_1 'Star Wars VII' and e_3 'J.J. Abrams' (DIRECTED BY) can be extracted from this example sentence.

We use the Wikidata schema to define the set of relations to extract in our experiments: $R = \{\oslash\} \cup \{r | r \in R_{wikidata}, \langle x, r_{x,y}, y \rangle, x, y \in \{\text{entity}, \text{number}, \text{date}\}\}$. We do not restrict the domain and take all relations types that are defined for pairs of entities, numbers and dates. Wikidata has only binary relations and each of them always connects an entity to another object. We take the relation types with the second position filled by an object of the type entity, number or date. We thereby exclude relation types that connect entities to external links, identifiers, file objects etc, since those are uncommon in texts. Table 3.1 lists some of the frequent Wikidata relation types, and we refer to Section 2.1 for more information on the Wikidata schema.

For the relation extraction task, we disregard the directionality of the relation and consider $\langle e_1, r, e_2 \rangle$ and $\langle e_2, r, e_1 \rangle$ to be the same relation instance. Wikidata includes an ontology and each entity belongs to one or more classes. The relation type definitions specify the Wikidata classes both for the left and right arguments. We can safely ignore the directionality for the purposes of the relation extraction experiments in this section, as it can be derived from the entity classes and the relation definition if needed for the downstream task.

For the relation extraction methods to be successful, it is crucial to extract relevant features from the sentential context (Riedel et al., 2010; Zeng et al., 2015). Standard approaches focus just on the relation between the target entities and disregard other relations that might be present in the same sentence (Zeng et al., 2015; Lin et al., 2016). For example, to correctly identify the relation type between the movie e_1 'Star Wars VII' and the director e_3 'J.J. Abrams' in the example in Figure 3.2, it is

Relation type	Example
country	⟨Darmstadt:Q2973, country:P17, Germany:Q183⟩
located in	⟨Darmstadt:Q2973, located in:P131, Hesse:Q1199⟩
shares border	⟨Germany:Q183, shares border:P47, France:Q142⟩
instance of	⟨Germany:Q183, instance of:P31, federal state:Q43702⟩
sport	⟨Florence Griffith:Q31082, sport:P641, athletics:Q542⟩
citizenship	⟨Florence Griffith:Q31082, citizenship:P27, United States of America:Q30⟩
subclass of	⟨federal state:Q43702, subclass of:P279, federal system:Q22676603⟩
spouse	⟨Ada Lovelace:Q7259, spouse:P26, William King-Noel:Q4426480⟩
founded by	⟨European Union:Q458, founded by:P112, West Germany:Q713750⟩, ⟨European Union:Q458, founded by:P112, France:Q142⟩
cast member	⟨Lupita Nyong'o:Q3840847, cast member:P161, Star Wars: The Force Awakens:Q6074⟩

Table 3.1: Frequent Wikidata relation types and example relation instances.

important to separate out the instance_of relation between the movie and its type e_2 'space opera epic film'. A common way to limit the context for a single relation is to focus on the text between the target entities (Zeng et al., 2015). However, for the entity pair $\langle e_1, e_3 \rangle$ in Figure 3.2, the text between the entities includes the whole sentence with the other entity and the other relation instances. It would be thus insufficient to simply assume that the whole sentence expresses just a single relation or that all of the features and markers for different relations are equally distributed. We consider this particular problem in the context of the relation extraction task and propose a solution for extracting multiple relations from the same sentence. This allows to improve the relation extraction accuracy and build a foundation for our follow-up semantic parsing work.

3.1.2 Distantly Supervised Dataset

To facilitate our experiments and the future research into joint relation extraction, we construct a dataset for the English language that contains multiple positive and negative relation instances per single sentence. We have motivated the need for the methods that extract multiple relations, that is all available relations from the same sentence in the previous section. In the same section, we have used the Wikidata knowledge base to define the set of relation types for open-domain relation extraction and consequently, we employ Wikidata to build the new dataset. We have presented the Wikidata knowledge base in detail in Section 2.1. For the dataset creation, we rely on several important features of Wikidata. First, Wikidata has a diverse set of relations that are frequently used. Most of the relation types in

Wikidata are constantly used to add new facts to the knowledge base (Färber et al., 2015). This will enable us to create a dataset that features hundreds of diverse relations. Second, Wikidata encodes the world knowledge strictly in a form of binary relation instances (⟨Germany:Q183, capital:P36, Berlin:Q64⟩), which simplifies the dataset construction and the experimental setup (we need to predict relations for all entity pairs, but not for triples or other entity combinations). Third, there is little to no duplication of information in Wikidata. A broad community oversight, similar to Wikipedia, ensures a higher data quality compared to other knowledge bases (Färber et al., 2015). For instance, there are few relation types that are the reverse of another type. Although there is a reverse parent relation instance to every child relation, for many other relation types there is no reverse. The cast member relation connects a movie to an actor, but there is no corresponding reverse relation. Finally, Wikidata is tightly integrated with the large corpus of Wikipedia articles. There is an unambiguous one-to-one mapping between Wikidata entities and Wikipedia articles. Through this link it is possible to directly use the English Wikipedia as a basis for the dataset. Wikipedia offers a corpus of long sentences that contain multiple entities and facts (this was demonstrated in previous corpus creation efforts based on Wikipedia: Mintz et al., 2009; Ghaddar and Langlais, 2017; Thorne et al., 2018). Therefore, Wikipedia is well-suited to create a corpus for the joint extraction of knowledge base relations.

We use a distant supervision approach (Mintz et al., 2009) to construct the dataset. Distant supervision is a technique to align a set of facts from an external resource with the text without the use of manual annotations (Gurevych et al., 2016, Section 5.2). For every fact, we assume that if the two entities participating in the fact appear in the same sentence then that particular fact is expressed in the sentences. Many combinations of entities form only a single fact in the knowledge base, that is they are connected with only one relation type. If they co-occur in text, it is likely that their relationship is being described. Distant supervision uses this to label a lot of text with relation types automatically and then learn a model that would predict relation types for new entity pairs.

Mintz et al. (2009) and Riedel et al. (2010) have successfully applied distant supervision to create relation extraction datasets for large-scale knowledge bases before. Their data contains a single relation instance per sentence and their task setup is not aligned with the broad semantic parsing framework, but is rather aimed at the knowledge base completion (as we have discussed above). This is in contrast to the dataset with multiple relations per sentence that we are describing in this section.

Wikipedia and Wikidata are tightly integrated, which enables us to employ manual wiki annotations in Wikipedia to extract high quality Wikidata annotations. Since Wikidata entities are interlinked with the Wikipedia articles, we can use the Wikipedia data to find usage examples for Wikidata relations. Each entity in Wikidata has a backlink to the Wikipedia articles about this entity in every language where such an article exists. Hence there is a mapping from the Wikidata

Q-IDs[1] to Wikipedia article names: $Q \mapsto W$. The mapping is many to one, but we keep only one-to-one links and then reverse the mapping: $W \mapsto Q$. This enables us to map Wikipedia article names to Wikidata IDs.

The Wikipedia articles are interlinked with each other by means of wiki links placed inside the article text. Usually, the first reference to another article in the text is explicitly marked with the wiki annotation as a link to the corresponding article. For instance: "The Skellig Michael island is named after the archangel [[Michael (archangel)|Michael]], while *Skellig* is derived from the [[Irish language]] word *sceilig*, meaning a splinter of stone.", where [[...]] is the wiki format for links. The text inside [[ ]] is the name of the linked article. Using the extracted mapping $W \mapsto Q$, we can convert every link to the Wikidata ID and thus produce a document annotated with Wikidata entities.

Having the text annotated with Wikidata entity IDs, we use the distant supervision assumption to add relation type annotations. We construct a mapping from entity ID pairs to relation types: $\langle \text{entity}, \{\text{entity}, \text{number}, \text{date}\} \rangle \mapsto R$, where the order of the entity IDs matters. The second argument may also be a number or a date. This allows us to check if two entities are connected in the knowledge base and if they form a relation triple. If any two entities are connected with more than one relation type and thus would violate the mapping, we randomly pick one relation type, because the distant supervision assumption requires that there is an unambiguous mapping between entity pairs and relations. We find it to be rarely the case in practice and the most entity combinations are connected by a single relation in the knowledge base. For comparison, Surdeanu et al. (2012) similarly report that only 7.5 % of entity pairs have more than one corresponding relation type in the distantly supervised dataset of Riedel et al. (2010).

Figure 3.5 shows the main distant supervision steps, in particular the mapping of links in a Wikipedia article to Wikidata IDs. For every possible entity pair in the sentence, we check if it can be mapped to a relation type and add it as a relation annotation to the sentence. If the considered entity pair is not the part of the mapping, we take that these two entities are not related and add an empty relation $\oslash$ as an annotation. Both outcomes are depicted in Figure 3.5.

We used the latest Wikipedia dump of May 1, 2016 at the moment of the corpus construction. It contained more than 5 million English articles. We start with the complete English Wikipedia corpus and perform the following concrete steps to generate the dataset:

1. We pre-process each article and remove any mark up, lists and tables except for the wiki links.

2. For each article, we collect a mapping of text strings to wiki links: $S \mapsto W$ (for instance 'Star Wars' $\mapsto$ Q462 would be extracted from the article snippet

[1] Unique IDs for entities in Wikidata have a Q-prefix, e.g. Q2973 for Darmstadt: `https://www.wikidata.org/wiki/Q2973`.

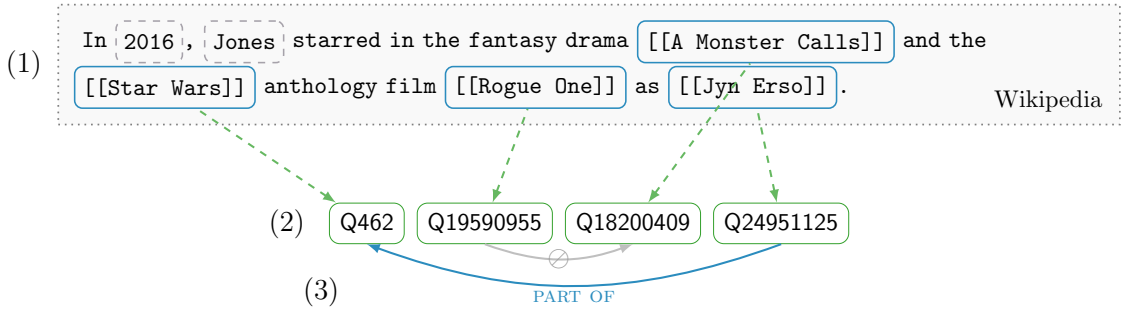

Figure 3.5: An example Wikipedia snippet with links to other articles and the main distant supervision steps. We (1) extract Wikipedia link mark-up, (2) convert it to Wikidata entity IDs, and (3) retrieve relations between them from Wikidata.

in Figure 3.5). This is necessary, since most of the time only the first mention of a particular entity is marked as a wiki link in an article. We go through the article and add the markup to the same text string as the first mention of the entity using the mapping $S \mapsto W$. We do not do co-reference resolution and ignore other references to the same entity in the text.

3. We split each article into a sentence set and perform further processing on the sentence level.

4. From each sentence, we extract the link markup and retrieve Wikidata entity IDs corresponding to the linked articles using the mapping $W \mapsto Q$ ((2) in Figure 3.5).

5. We remove sentences that are longer than 180 characters because of the computational restrictions of the noun chunker used in the next step.

6. We extract named entities and noun chunks from the input sentences with the Stanford CoreNLP toolkit (Manning et al., 2014) to identify entities that are not covered by the Wikipedia annotations (e.g. `Jones` in the example sentence in Figure 3.5). We retrieve IDs for those entities by searching through entity labels in Wikidata and looking for an exact label match. This increases the amount of entity mention annotation further.

7. We use the HeidelTime tool (Strötgen and Gertz, 2013) to extract dates. We identify numbers with a simple rule.

8. We filter out sentences that contain fewer than 3 annotated entities, since our goal is to have multiple relations per sentence.

9. We build a set of all possible entity pairs in the sentence: $P = \{\langle e, x\rangle | e \in E_{wikidata}, x \in \{E_{wikidata}, N, D\}\}$, where the second position can be filled by an entity, number or a date.

10. For each pair in P, we check the mapping $\langle E, \{E, N, D\}\rangle \mapsto R$ for a relation that connects the entity pair or the entity with a number/date. If there is a relation, we add the pair and the corresponding relation type as annotation

	Train	Validation	Held-out
# of sentences	372,059	123,824	360,334
# of unique relation triples	284,295	113,852	287,902
# of relation instances	777,481	252,277	740,963
of those positive	548,587	178,531	573,881
of those negative	228,894	73,746	167,082

Table 3.2: Statistics of the generated Wikidata relation extraction dataset from Wikipedia.

to the sentence. If there is no mapping for the given pair, we add the empty relation $\oslash$ as annotation ((3) in Figure 3.5). The latter serves as a negative instance for the subsequent model training (see Section 3.1.4).

11. Finally, we only retain sentences that have at least one positive relation instance.

The constructed dataset features 353 different relation types (out of approximately 1700 relation types in the Wikidata schema that satisfy conditions for the set R that we defined above). We split it into train, validation and held-out sets, ensuring that there is no overlap in either sentences or relation triples between the three sets. Table 3.2 summarizes the statistics about the dataset.

The dataset is freely available under the CC-BY-SA 4.0 license at the UKP lab website in JSON format[2]. Each file is a list of objects that correspond to sentences (see Listing 3.1). We include with each sentence: the tokenized sentence content, the annotated relations with Wikidata IDs and the list of entities that participate in these relations.

We have assessed the quality of the distant supervision setup on 200 manually verified sentences from the training set: 79.5 % of relations in those sentences were correctly labeled with distant supervision (86.9 % for entity-number and entity-date relations, 74.7 % for entity-entity pairs). This outsteps the approximately 70 % accuracy for a distant supervision dataset previously reported by Surdeanu et al. (2012). We attribute the improvement over the previous dataset to the quality of the information in Wikidata and the fact that the Wikipedia link markup can be directly translated into the Wikidata entity annotations, which eliminates entity linking errors.

3.1.3 Approaches to Relation Extraction

Multiple relations co-occur in the same sentence frequently in the NLP data: for instance, 30 % of the questions in the question answering data (Berant et al., 2013) and 20 % of facts in the fact verification dataset (Thorne et al., 2018). Yet, common

[2] https://www.informatik.tu-darmstadt.de/ukp/research_6/data/lexical_resources/wikipedia_wikidata_relations/

```
{
  "vertexSet": [
    {"kbID": "Q15284", "tokenpositions": [4]},
    {"kbID": "Q12995", "tokenpositions": [9]},
    {"kbID": "Q31", "tokenpositions": [11]},
    {"kbID": "Q15253706", "tokenpositions": [6, 7]},
    {"kbID": "Q3099657", "tokenpositions": [0]}
  ],
  "edgeSet": [
    {"left": [6, 7], "right": [4], "kbID": "P279"},
    {"left": [0], "right": [11], "kbID": "P17"},
    {"left": [0], "right": [9], "kbID": "P361"},
    {"kbID": "P0", "right": [11], "left": [9]}
  ],
  "tokens": ["Marke", "is", "a", "sub", "municipality", "of",
      "the", "city", "of", "Kortrijk", ",", "Belgium", "."]
}
```

Listing 3.1: A snippet of the Wikipedia-Wikidata relation extraction dataset in the JSON format. Here a single sentence is shown, we use 'edgeSet' to store relations and 'vertexSet' to store information on the entity mentions.

relation extraction methods assume only one relation per sentence (Zeng et al., 2015). Our methodological contribution to relation extraction is focused on using the co-occurrences of relations in the same sentence to improve the extraction of the correct relation type for each of those relation instances.

From the task perspective, we treat each relation instance as a relation classification task: predict a relation for each target based on the sentential context. The previous relation extraction work focuses mostly on predicting a single relation type based on the combined evidence from all of the occurrences of an entity pair, that is extracting new relations between entities based on an accumulation of facts. Compare the task description and our discussion of Riedel et al. (2010); Zeng et al. (2015); Lin et al. (2016) in Section 3.1.1.

Regardless of the actual task formulation, the aforementioned body of work presents multiple methods for extracting features from the input to effectively predict semantic relations. We discuss the feature extraction below.

Mintz et al. (2009) and Riedel et al. (2010) have been influential for the last decade of relation extraction research by establishing the framework for the task and applying the distant supervision paradigm at scale for the first time. Distinguishing about each paper is the usage of the largest available world knowledge base (Freebase at that time) and the introduction of new relation extraction benchmarks. We have covered the distant supervision technique in the above section and described its application to construct the dataset for our use case.

In our approach, we predict the target relation from a feature vector: we employ a neural network to automatically encode the target relation and the sentential context into a fixed-size feature vector. Mintz et al. (2009) and Riedel et al. (2010) have used manually engineered features based on part-of-speech tags and dependency parses to represent the target relations. Those were outperformed by subsequent neural approaches (Zeng et al., 2015; Zhao et al., 2015; Lin et al., 2016). Zeng et al. (2015) and Zhao et al. (2015) have specifically shown that one can successfully apply convolutional neural networks to extract sentence-level features for relation extraction without residing to manual feature definitions.

All of the previous approaches consider just the relation between the target entities and disregard other relations that might be present in the same sentence. Co-occurring relation instances in the same context are the novelty of our method. One can also use other types of structured information from the nearby context to improve relation extraction. Roth and Yih (2004) have combined Named Entity Recognition (NER) and relation extraction in a structured prediction approach to improve both tasks. Their method uses linear programming to formulate mutual constraints that relation extraction and NER pose for each other, that is that the relation is to be recognized between the identified entity mentions and not any other pair of tokens. Li et al. (2013) have designed global features and constraints to extract multiple events and their arguments (a task similar to entity and relation extraction) from the same sentence without linear programming. In a similar manner, Miwa and Bansal (2016) have implemented an end-to-end neural network to construct a context representation for joint entity and relation extraction. Their network does not explicitly encode the constraints between entity mentions and relations, but they are learned in the latent representations inside the network.

We do not implement global constraints in our approach, since unlike events and arguments, there are no restrictions as to what relations can appear together. Instead, we follow the neural approaches of Miwa and Bansal (2016) and Lin et al. (2016) and learn latent context representations. We encode all relations in the same context into fixed-size vectors, combine them and use this information to make the final prediction.

3.1.4 Context-Aware Model Architecture

We present a novel architecture that considers other relations in the sentence as a context for predicting the label of the target relation. We create rich context representations by computing relation vectors for each relation instance in the sentence and combining them. We aim to learn correlations and restrictions that different relations may impose on each other. In practice, some relation types may tend to co-occur, such as DIRECTED_BY and PRODUCED_BY, whereas others may be restrictive (e. g. one can only have a single PLACE_OF_BIRTH).

We use the term *context relations* throughout this section to refer to all the other

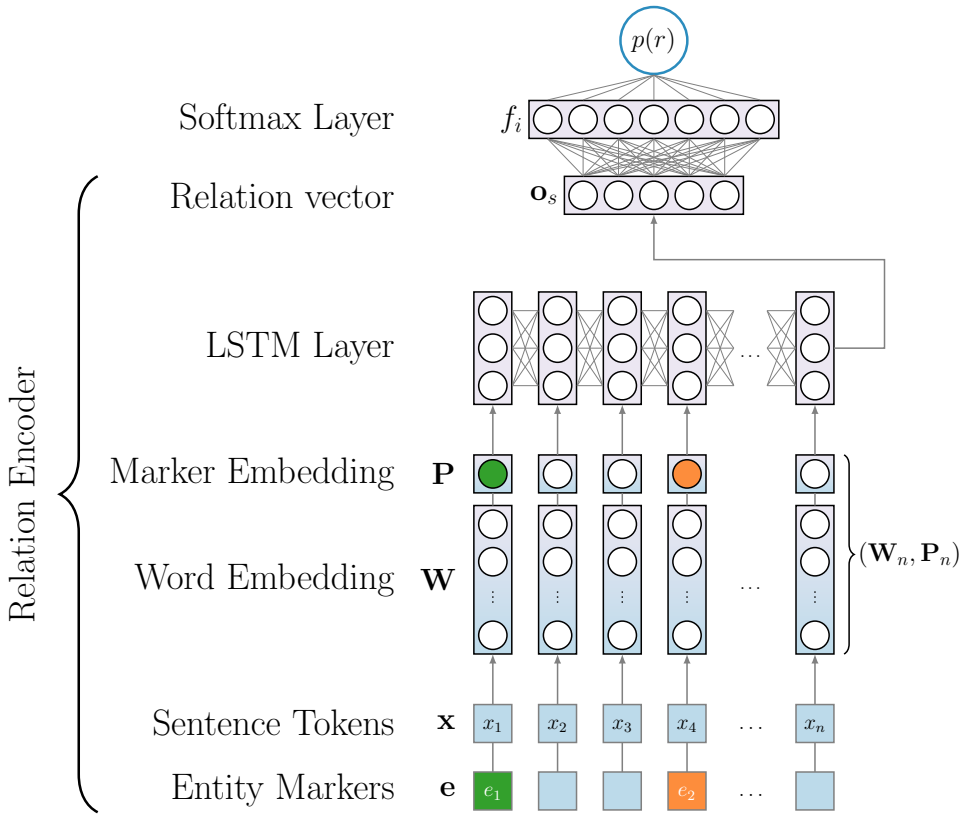

Figure 3.6: The architecture of the relation encoder for an entity pair $\langle e_1, e_2 \rangle$. We encode the tokens and the entity positions with the embedding matrices W and P. The token and marker embeddings are concatenated and fed through an LSTM layer to compute a relation vector. This vector encodes the sentential context for the specified pair of entities.

relations in the sentences but the target relation. We learn representations for all relations in a single sentence and combine the representation of the target relation and representations of the context relations to make the final prediction.

Relation encoder At the center of our model architecture is the relation encoder, which is applied to extract features for relation mentions in text: both the context and the target relations. The relation encoder produces a fixed-size vector representation $\mathbf{o}_s$ of a relation between two entities in a sentence (see Figure 3.6).

First, each token of the sentence $\mathbf{x} = \{x_1, x_2 \dots x_n\}$ is mapped to a k-dimensional embedding vector using a matrix $\mathbf{W} \in \mathbb{R}^{|V| \times k}$, where $|V|$ is the size of the vocabulary. For the relation extraction experiments, we use 50-dimensional GloVe embeddings pre-trained on a 6 billion corpus (Pennington et al., 2014).

Second, we mark each token in the sentence as either belonging to the first entity e_1, the second entity e_2 or to neither of those (see the bottom of Figure 3.6). A marker embedding matrix $\mathbf{P} \in \mathbb{R}^{3 \times d}$ is randomly initialized (d is the dimension of the position embedding and there are three marker types). For each token, we concatenate the marker embedding with the word embedding: $(\mathbf{W}_n, \mathbf{P}_n)$. This

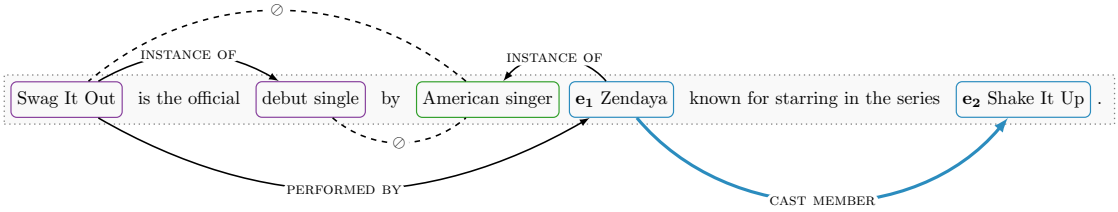

Figure 3.7: The target relation between the entity mentions e_1 and e_2 (in blue) and the other context relations in the same sentence.

creates a unique input sequence for each entity pair in each sentence (cf. Figure 3.8). For different relations between different entity pairs in the same sentence, the word embeddings would be the same but the positional markers would be different denoting that each time we are interested in a different relation in the same input text.

We apply a Recurrent Neural Network (RNN) on the concatenated token embeddings. The length n naturally varies from sentence to sentence and an RNN provides a way to accommodate inputs of variable sizes. It maps a sequence of n vectors to a fixed-size output vector $\mathbf{o}_s \in \mathbb{R}^o$. We take the output vector $\mathbf{o}_s$ as the representation of the relation between the target entities in the sentence (see the top of Figure 3.6). We use the Long Short-Term Memory (LSTM) variant of an RNN (Hochreiter and Schmidhuber, 1997) that was successfully applied to information extraction before (Miwa and Bansal, 2016).

Our hypothesis is that the context relation information is useful to predict the target relation and that ultimately predicting jointly multiple relations appearing in the same sentence jointly leads to an improved relation extraction system. We set up our experiments to test it. Our main idea to incorporate the co-occurrence of the relations into the network is agnostic to the particular sentence encoding architecture. We use the encoder network described above to test our hypothesis and derive three models from it: a no-context relation encoder baseline, a ContextSum model that uses a simple sum to aggregate relation vectors and a ContextWeighted model that adds weights to the aggregation process.

Relation encoder baseline As the first model variant, we feed the output vector $\mathbf{o}_s$ of the relation encoder to a softmax layer to predict the final relation type for the target entity (see the top of Figure 3.6):

$$p(r|\langle e_1, e_2\rangle, \mathbf{x}; \theta) = \frac{\exp(f_r)}{\sum_{i=1}^{n_r} \exp(f_i)}, \qquad (3.1)$$
$$f_i = \mathbf{y}_i \cdot \mathbf{o}_s + b_i,$$

where $\mathbf{y}_i$ is a weight vector and b_i is a bias.

ContextSum model We argue that for predicting a relation type for a target entity pair other context relations in the same sentence are relevant. Therefore, we propose two variants of an architecture that in addition to the target entity pair feature vector incorporates features for the other entity pairs from the same sentence: **ContextSum** and **ContextWeighted**. The dataset that we have constructed in Section 3.1.2 includes multiple entities and relations per sentence and we utilize this to construct the set of context relations. We have considered all possible entity pairs in the sentences when constructing the dataset and we take all of them as context relations. In practice, we break down large sets of context relations into batches of 7 to speed up the computation.[3] Figure 3.7 shows the target relation for the entity pair $\langle e_1, e_2 \rangle$ and the context relations.

We apply the same relation encoder on the target and context relations (see Figure 3.8). That ensures that the representations for target and context relations are learned jointly. We sum the context relation representations: $\mathbf{o}_c = \sum_{i=0}^{m} \mathbf{o}_i$, where each element $\mathbf{o}_i$ is a vector representation of a single context relation and m is the number of context relations in the sentence. The resulting context representation $\mathbf{o}_c \in \mathbb{R}^o$ is concatenated with the vector representation of the target relation: $\mathbf{o} = [\mathbf{o}_s, \mathbf{o}_c]$. We feed the concatenated vector to the softmax layer in Eq. 3.1 to predict the final relation type for the target entity pair (see the upper part of Figure 3.8).

ContextWeighted model For the second variant, we use a weighted sum of the context relation representation at the penultimate step:

$$\mathbf{o}_c = \sum_{i=0}^{m} a_i \mathbf{o}_i, \quad a_i = \frac{\exp(g(\mathbf{o}_i, \mathbf{o}_s))}{\sum_{j=0}^{m} \exp(g(\mathbf{o}_j, \mathbf{o}_s))}, \tag{3.2}$$

where g_i computes a weight for a context relation with respect to the target relation: $g(\mathbf{o}_i, \mathbf{o}_s) = \mathbf{o}_i \mathbf{A} \mathbf{o}_s$, and $\mathbf{A}$ is a weight matrix that is learned.

Training the models

All model variants were trained using the Adam optimizer (Kingma and Ba, 2014) with categorical cross-entropy as the loss function. We use an early stopping criterion on the validation data to determine the number of training epochs. The learning rate is fixed to 0.01 and the rest of the optimization parameters are set as recommended in Kingma and Ba (2014): $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-08$. The training is performed in batches of 128 instances.

We apply Dropout (Srivastava et al., 2014) on the penultimate layer and on the embeddings layer with a probability of 0.5. We choose the size of the layers (LSTM layer size $o = 256$) and entity marker embeddings ($d = 3$) with a random search on the validation set. We have tested the values $\{64, 128, 256, 512\}$ for the LSTM layer size, the values $\{1, 3, 5, 7\}$ for entity marker embeddings and the values in the range 0.0–0.75 for the Dropout rate.

[3] 88 % of the sentences in our relation extraction corpus have fewer than 7 relation pairs.

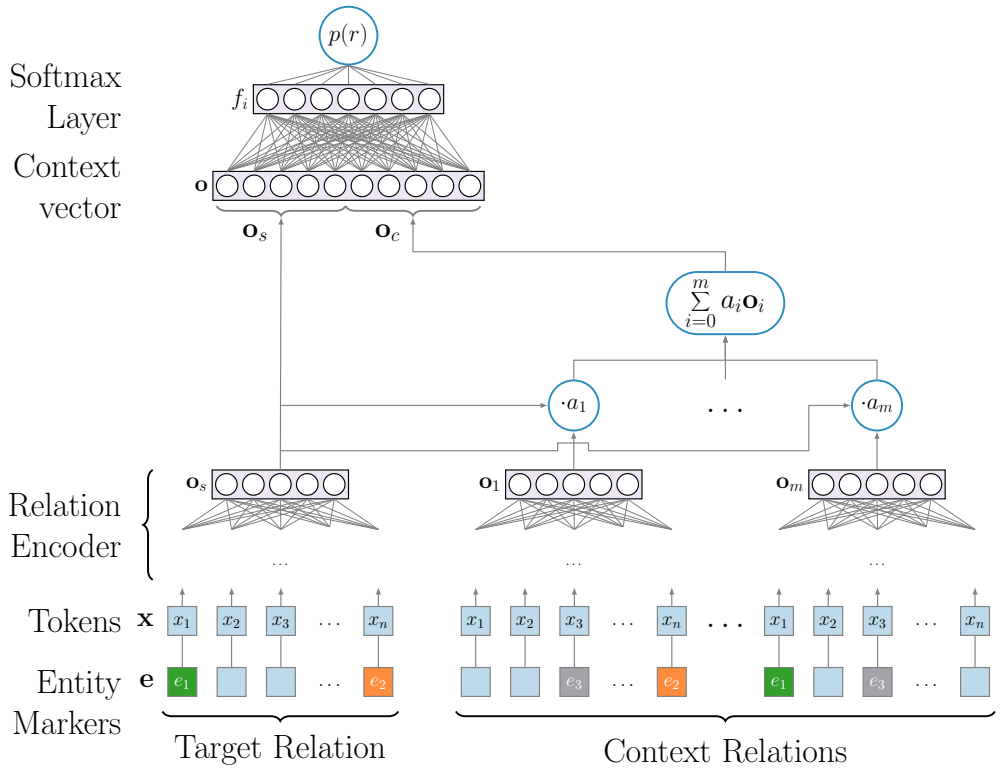

Figure 3.8: The full context-aware model architecture for relation extraction. The context relations are processed with separate relation encoders on the right and their relation vectors are incorporated through the weighted average vector $\mathbf{o_c}$. For the ContextSum model variant, $a_i = 1$.

3.1.5 Evaluation Methodologies

As an additional baseline, we re-implement a sentence-level model based on Convolutional Neural Networks (CNNs) described in Lin et al. (2016). This is a state-of-the-art model for fine-grained relation extraction that was previously tested on the single-relation dataset from Riedel et al. (2010). In addition to CNNs, their architecture uses a different position encoding scheme: position markers encode a relative position of each word with respect to the target entities. This is required, since in comparison to LSTM the CNN model has less ability to capture global positional information. We have also experimented with such richer position markers for our LSTM-based models during the development, but found no improvements and hence did not include them in the final experiments. We use the same GloVe word embeddings for the CNN model and perform a hyperparameter optimization on the validation set. We do not directly compare against the approach of Surdeanu et al. (2012) that also performs sentence-level relation extraction using manually defined features. The provided implementation for their system does not feature the complete pipeline and is only applicable on a particular Freebase dataset.

Our dataset lets us compare the baseline models and the models that use context relations on the same data. Following the previous work on relation extraction

(Mintz et al., 2009; Riedel et al., 2010), we report the aggregated precision-recall curves for each model on the held-out data (Figure 3.9). To compute the curves, we rank the predictions of each model by their confidence and traverse this list top to bottom measuring the precision and recall at each step s:

$$\text{Precision}(s) = \frac{T_p^{D[0:s]}}{T_p^{D[0:s]} + F_p^{D[0:s]}} \tag{3.3}$$

$$\text{Recall}(s) = \frac{T_p^{D[0:s]}}{T_p^{D[0:s]} + F_n^{D[0:s]}} \tag{3.4}$$

$$T_p^{D[0:s]} = \sum_{i=0}^{s} 1[r_i^p = r_i^t \wedge r_i^t \neq \oslash] \tag{3.5}$$

$$F_p^{D[0:s]} = \sum_{i=0}^{s} 1[r_i^p \neq r_i^t \wedge r_i^p \neq \oslash] \tag{3.6}$$

$$F_n^{D[0:s]} = \sum_{i=0}^{s} 1[r_i^p \neq r_i^t \wedge r_i^p = \oslash], \tag{3.7}$$

where $D[0:s]$ denotes a subset of the test set that includes the top s elements ranked by model confidence, r_i^t is the true relation type for the instance i and r_i^p is the model prediction for the same instance i. We compute the precision and recall treating every relation type as a positive class and the $\oslash$ as the negative class (micro-averaging).

We also include a precision-recall curve at the macro level, where we compute the precision and recall for each relation type separately at each step and then average the result (Figure 3.10). The macro curve does not overemphasize the relation types based on their frequency and shows which model performs best across all relation types.

3.1.6 Experimental Results

Taking only high confidence predictions and ignoring the rest gives small recall values, but high precision among the few predictions that remain. Conversely, taking all predictions disregarding the model confidence gives the maximum recall for the given model at the expense of the precision.

In Figure 3.9, we see that the models that take the context into account perform similar to the baselines at the smallest recall numbers, but start to positively deviate from them at higher recall rates. In particular, the ContextWeighted model performs better than any other system in our study over the entire recall range. In terms of the overall accuracy compared to the competitive LSTM-baseline that uses the same relation encoder, the ContextWeighted model achieves a 24 % reduction of the average error: from 0.2096 ± 0.002 to 0.1590 ± 0.002. The difference between the models is statistically significant ($p = 0.009$). We estimate

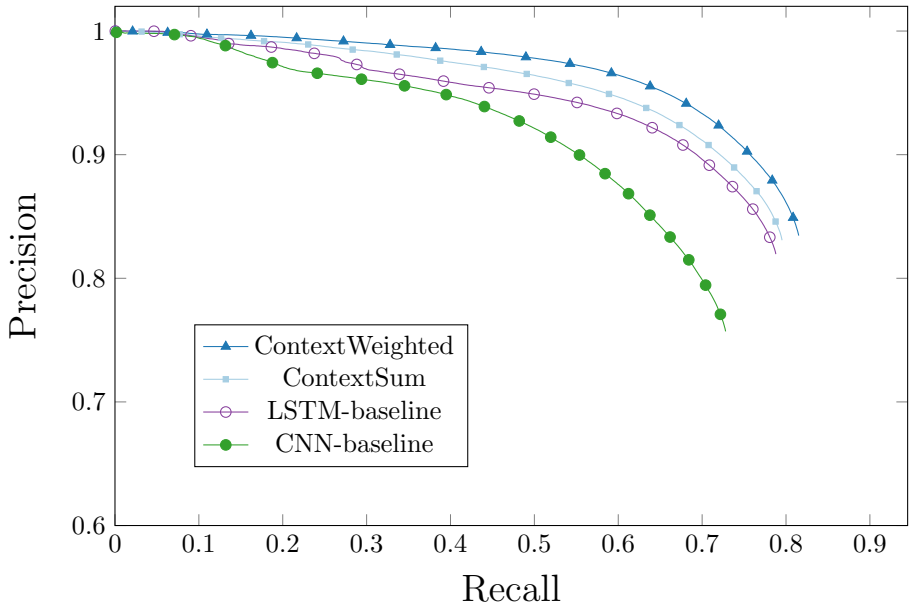

Figure 3.9: Aggregated precision-recall curves for the context-aware relation extraction models and the baselines.

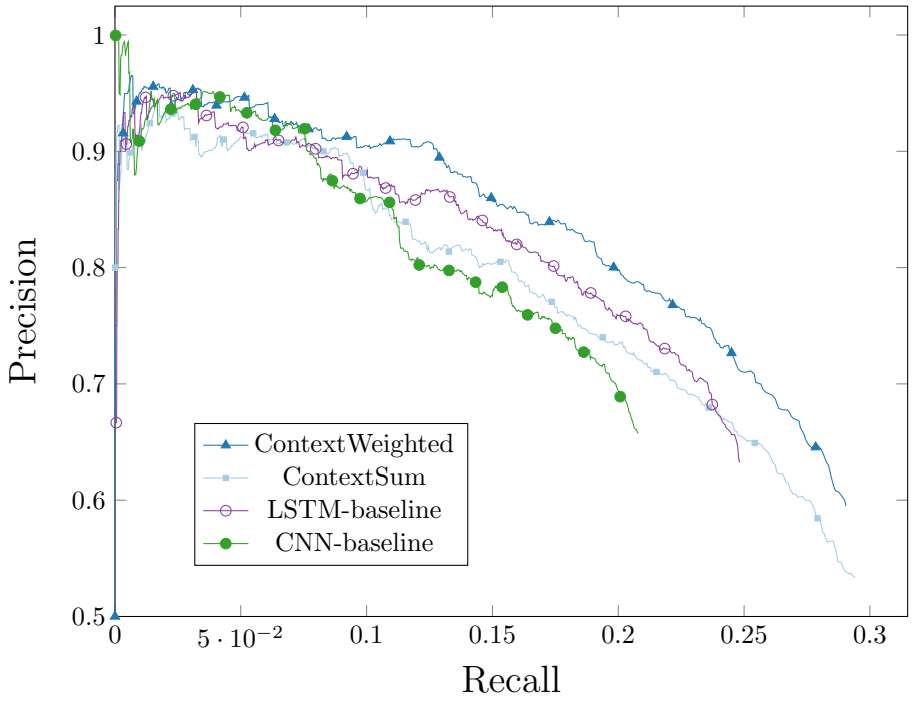

Figure 3.10: Aggregated macro precision-recall curves for the context-aware relation extraction models and the baselines.

the average error and the standard deviation on 5 training iterations for each model. The statistical significance is computed using the Wilcoxon rank-sum test[4] on the error rates.

[4] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ranksums.html`

Relation type	Frequency	LSTM-Baseline *P*	LSTM-Baseline *R*	ContextWeighted *P*	ContextWeighted *R*
country:P17	0.118	0.8899	0.9344	**0.9130**	0.9382
located in:P131	0.082	0.8329	0.8832	**0.8655**	**0.8994**
shares border:P47	0.048	0.7579	0.7078	**0.7962**	**0.8075**
instance of:P31	0.048	0.7864	**0.8568**	**0.8478**	0.8401
sport:P641	0.046	0.9753	0.9828	0.9822	0.9823
citizenship:P27	0.028	0.9001	0.9448	0.9041	0.9417
part of:P361	0.019	0.5623	0.4854	**0.6269**	**0.5113**
subclass of:P279	0.016	0.5230	0.4390	0.5272	**0.5908**

Table 3.3: Precision (*P*) and recall (*R*) of the ContextWeighted model for the top relations and their relative frequency in the Wikipedia-Wikidata dataset.

Dataset errors	
Annotation errors	13 %
Ambiguous instances	19 %
Model errors	
instance of vs. subclass of	4 %
Other wrong model predictions	64 %

Table 3.4: Manual error analysis on 100 test instances for the ContextWeighted model.

The macro precision-recall curves give equal weights to all relations in the dataset. Figure 3.10 shows that the ContextWeighted model performs best over all relation types. One can also see that ContextSum doesn't universally outperform the LSTM-baseline. It demonstrates in particular that using the learned weighting scheme is crucial to extract relevant information from the context relations and the simpler ContextSum architecture is not always sufficient.

3.1.7 Error Analysis

We break down the performance of the best context-aware model (ContextWeighted) and of the LSTM-baseline for the top relations in the dataset (Table 3.3). On the relation-specific results, we observe that the ContextWeighted model demonstrates the most improvement on precision, with recall being almost the same for the most relation types. The context-aware model seems to be especially useful for taxonomy relations (see subclass of, part of), whereas the performance on the sport and citizenship relation types does not differ between the models.

We have also manually inspected the model performance on 100 test instances. The findings are summarized in Table 3.4. We confirm again that the dataset

contains a certain number of annotation errors (13 %) due to the noisy distant supervision process (cf. manual inspection at the end of Section 3.1.2). 19 % of the errors are caused by ambiguous contexts or too broad definitions of the relations. For example, the distinction between an AUTHOR and a COMPOSER is not well defined in Wikidata. The rest of the errors (68 %) in the manually inspected instances are the actual wrong relation type predictions. Among the wrong model predictions, the largest group of errors is due to the model confusing the INSTANCE OF and the SUBCLASS OF relation types (4 %).

In the first part of this chapter, we have defined the relation extraction in the context of semantic parsing and have in particular contributed to the extracting multiple relations from the same context, the use case which we show is relevant for the downstream tasks. Relation extraction is a fundamental task for linking texts to knowledge bases and is conceptually one of the two main building blocks of grounded semantic parsing. Earlier, in Section 2.2 we have discussed that disambiguating and linking entities and relations is the central step on the way from an ungrounded to grounded sentence representation (see Figure 2.11).

3.2 Entity Linking

Linking entities in the input to an external knowledge base is the essential step of the grounded semantic parsing pipeline and consequently of the most knowledge-based methods (see the discussion in Section 2.2.3). Given the RQ1 formulated in Chapter 1, we are interested in effective methods for entity linking and through RQ2 and RQ3 — in how entity linking performs on the domain specific data for the downstream applications.

We again consider the case of the question answering downstream task to define requirements for our methods in this section. Question answering data features entities of various categories and is normally noisy, so that common features used in the conventional entity linking systems, such as capitalization, are not available. In this section, we investigate entity linking on question answering data and successfully tackle the aforementioned challenges.

We propose a jointly optimized neural architecture for entity mention detection and entity disambiguation. This ensures that any token n-gram can be considered as a potential entity mention, which is important to link entities of different categories, such as movie titles and organization names. Moreover, we model the surrounding context on different levels of granularity to help with the noise in the data. At the token level, we extract features from the whole question context, whereas at the character level, we only consider the candidate n-gram. Simultaneously, we extract features from the knowledge base context of the candidate entity: character-level features are extracted for the entity label and higher-level features are produced based on the entities surrounding the candidate entity in the knowledge base. This information is aggregated and used to predict whether the n-gram is an entity mention and to what entity it should be linked.

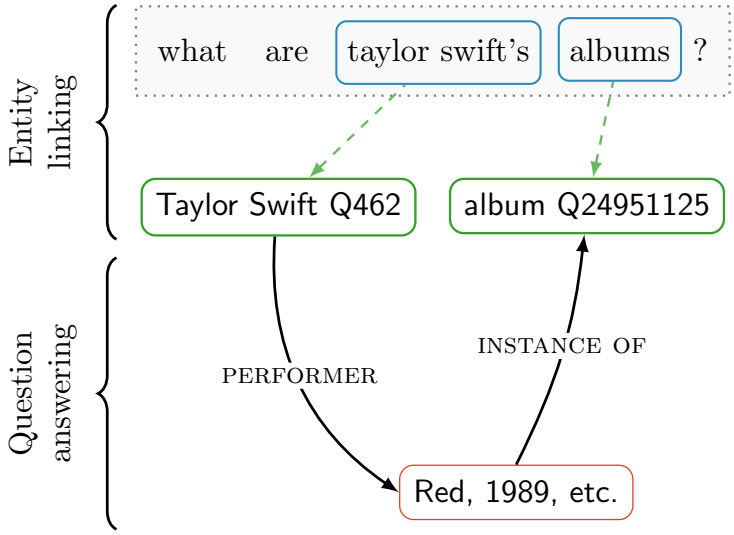

Figure 3.11: Entity linking for question answering. The example shows the effect that the two extracted entities have on the downstream question answering task. The upper part shows two entities in the question. In the lower part, this information is compared against the reference in the knowledge base to find an answer entity that is connected to the linked entities in the question.

We use the Wikidata knowledge base and available question answering datasets to create benchmarks for entity linking on question answering data for English. In the empirical evaluation, our approach outperforms the previous state-of-the-art system on this data, resulting in an average 8 % improvement of the final score.

3.2.1 Task Description

Entity linking is the task of identifying entity mentions in an input text and linking them to entities in a knowledge base. For example, two entity mentions are detected and linked to the knowledge base referents in Figure 3.11. We give an example from two downstream tasks. The result of the entity linking is crucial for the downstream applications, such as question answering or fact checking, that rely on the identified mentions to pull the relevant information. For instance, question answering systems operate under the assumption that the correct answer must be connected via some path over the knowledge base to the entities mentioned in the question. Otherwise, there is no way to arrive at the answer starting from the input question. Grounded semantic parsing assumes in general that all entities in the representation are to be linked to their referents.

Several entity linking systems are freely available and can be readily applied. DBpedia Spotlight[5] is a popular system for recognizing DBpedia entity mentions in texts and AIDA[6] was developed for linking texts to Wikipedia articles. These

[5] http://www.dbpedia-spotlight.org
[6] https://www.mpi-inf.mpg.de/yago-naga/aida/

systems were developed and trained for long documents, such as news texts, and are less suited for short and noisy data that is more typical of the NLP applications regarded in this thesis. We use the existing systems as a basis for our approach to entity linking and as a point of comparison. We follow Hoffart et al. (2011) and Mendes et al. (2011) in particular and formulate the entity linking task as follows. We consider an input utterance $\mathbf{x} = [x_1 \dots x_n]$ that consists of n tokens. The two subtasks follow: mention detection and entity disambiguation.

Mention detection produces a set of token spans from the input $\mathbf{x}$ that are possible references to world entities: $M = \{m_i \dots m_j\}$. It depends on the knowledge base, the inventory of the world entities available and the definition of an entity mention, if a particular token span should be considered an entity mention. The token spans produced during the mention detection step may be allowed to overlap or be strictly non-overlapping depending again on the inventory of the entities and on the downstream task. For instance, the whole phrase "president of the United States" may be linked to the Wikidata entity President of the United States:Q11696 and in the same phrase the token spans 'president' and 'the United States' may be linked to the separate entities president:Q30461 and United States of America:Q30. If just linking the longest token span 'president of the United States' would be sufficient, may be decided only considering the downstream task requirements. That is one of the reasons why we consider the entity linking task in the context of question answering in this work. The final decision, if a particular span m is an entity mention or not is often deferred to the next subtask.

Entity disambiguation considers possible referents in the knowledge base for the identified spans m and maps each of them to a single entity. A mention span can be ambiguous and can potentially refer to multiple entities, akin to the classical word disambiguation problem. Typically, the knowledge base is consulted for possible entity disambiguation candidates for a particular candidate token span (e.g. by means of a full text search).

For example, in Figure 3.12 the mention detection step would possibly produce a set of spans: m = [George Clooney, CEO, Apple]. The task of the entity disambiguation component is to retrieve the entity linking candidates for each detected mention span and to disambiguate the spans if more than one candidate is available. So 'Apple' might be referring to the computer company, a music label, a music album or a fruit, whereas 'George Clooney' is an unambiguous mention if we consider entity candidates from Wikidata. Depending on the downstream task, the entity disambiguation component may be allowed to link the detected mention to an empty candidate and thus correct the mention detection mistake or disregard it as irrelevant. In this example, it might be useful to link the mention 'CEO' to a type of an administrative position as described in Wikidata or conversely, leave it unlinked, since in the example sentence "George Clooney is a CEO of Apple" it rather describes a relation between the two entities — something that would be covered by the relation extraction task. A similar case is the extraction and linking of 'album' in the example in Figure 3.11. To answer the

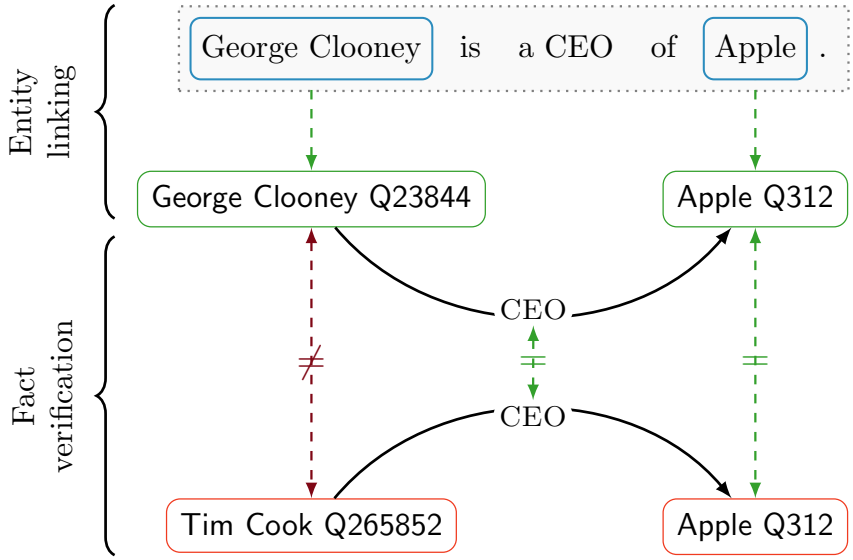

Figure 3.12: Entity linking for fact checking. The upper part shows two entity mentions that are then compared against the reference knowledge in a knowledge base (the lower part).

question correctly, it is useful to link 'album' to a type of a musical record and use it to restrict the returned answer set.

From the above example, we can see that the type of the entities that should be extracted and linked is highly dependent on the downstream tasks and its requirements. We discuss the question answering task and what kind of challenges arise with entity linking on short text data typical for the knowledge base question answering task in the next section.

3.2.2 Challenges for Entity Linking on Question Answering Data

Several benchmarks exist for entity linking on Wikipedia texts and news articles, such as ACE (Bentivogli et al., 2010) and CoNLL-YAGO (Hoffart et al., 2011). These datasets contain multi-sentence documents and cover three categories of entities: Location, Person and Organization. These categories are commonly recognized by NER systems, for instance, by the Stanford NER Tool (Manning et al., 2014), which is commonly used for pre-processing in entity linking tools. As a consequence, the existing entity linking systems are better applicable to long well-formed texts and are less suited for short and noisy data that is more common in the question answering scenarios. Moreover, they are limited to recognizing the three aforementioned entity categories.

Let us consider question answering, the primary downstream application for our semantic parsing framework. We have seen above that extracting other categories of entities such as professions (e.g. 'president') or product types (movie titles or songs) is required for question answering. Moreover, question answering datasets are normally collected from the web and contain noisy and diverse data (e.g. the

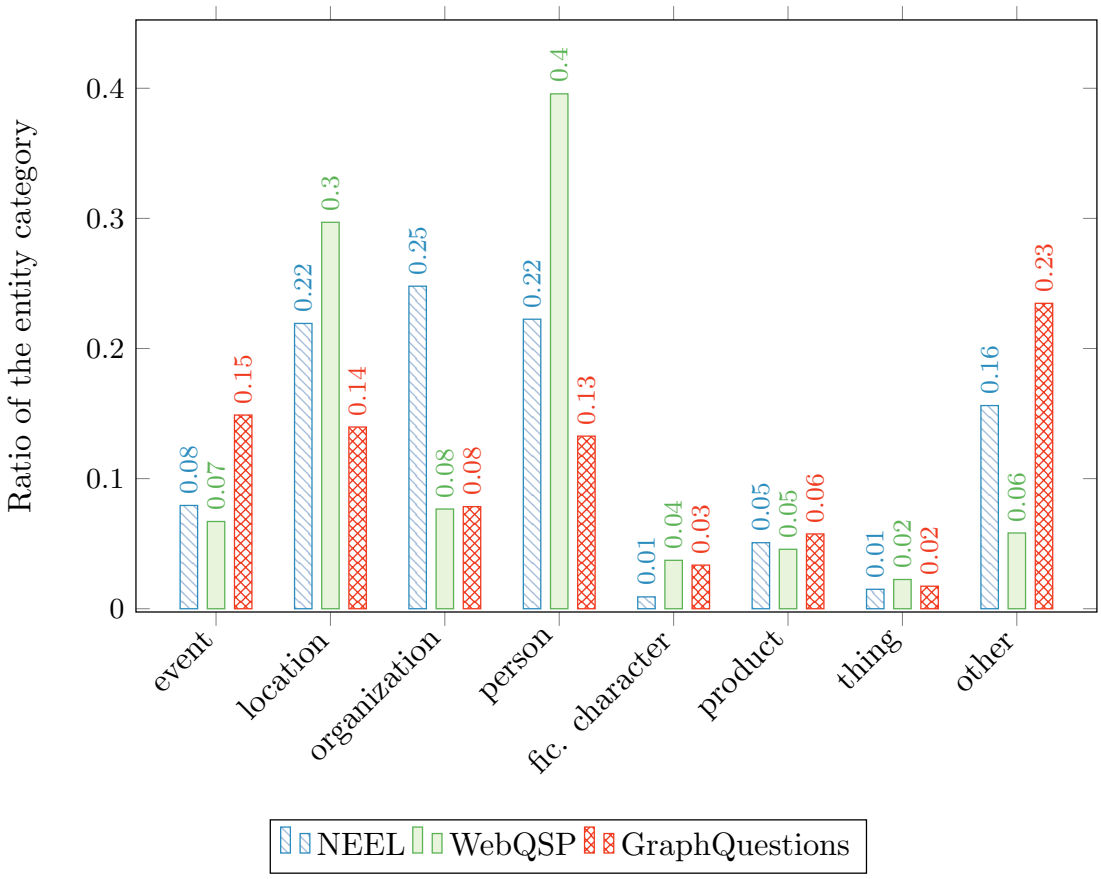

Figure 3.13: Distribution of entity categories in the NEEL 2014, WebQSP and GraphQuestions datasets.

WebQuestions dataset of Berant et al., 2013). This poses a number of challenges for the entity linking systems developed with common Wikipedia datasets. First, common features used in entity linking models, such as capitalization, are not available on the noisy data. Second, a short text snippet, such as a web query question, does not contain a broader document context for joint entity matching and similar techniques (Moro et al., 2014).

In the recent years, entity linking on Twitter data has emerged as a new branch of entity linking research. Tweets are challenging short pieces of text that often contain abbreviations and usually exhibit a spoken-like vocabulary (Ritter et al., 2011). In that regard tweets are similar to the typical question answering data. Twitter data is also directly relevant for the fact checking downstream application, since many information verification efforts focus on Twitter (Thorne and Vlachos, 2018). Entity linking on tweets was the central task of the NEEL shared task from 2014 to 2016 (Rizzo et al., 2017). Yet, as we will see below, the existing Twitter entity linking datasets are limited with respect to the entity categories covered.

We have already highlighted the fact that the existing freely available entity linking systems focus solely on three named entity categories: Location, Organization and Person. Question answering, fact verification and other downstream applications

often have to deal with a much more diverse distribution of entities. For instance, movie titles are a common entity categories for question answering on the web. One can imagine numerous queries regarding cast members or the release date of a movie. This renders the traditional freely available entity linking systems, such as AIDA and DBpedia Spotlight, limited for question answering and fact verification.

In Figure 3.13, we examine three entity linking datasets with respect to the entity category distribution: the Twitter data from the NEEL shared task (Rizzo et al., 2017) and two knowledge base question answering datasets, WebQSP (Yih et al., 2016) and GraphQuestions (Su et al., 2016). The data for the NEEL shared task was annotated with 8 broad entity categories, that besides Location, Organization and Person include Fictional Character, Event, Product (such as electronic devices or works of art), Thing (abstract objects) and Other (e.g. professions). Figure 3.13 shows the distribution of entity categories in the training set from the NEEL 2014 competition. One can see on the diagram that the distribution is still mainly skewed towards 3 categories: Location, Person and Organization, which are the top three categories for the NEEL shared task dataset.

Figure 3.13 also shows the entity categories present in the two question answering datasets. The distribution over the categories is more diverse in this case. We can see that it differs from the Twitter data with respect to the entity categories covered. The WebQuestions dataset includes the Fictional Character and Thing categories, which are almost absent from the NEEL dataset and the Organization category is much less prominent. However, Location and Person are still the top categories in WebQuestions. A more uniform distribution can be observed in the GraphQuestion dataset that features many entities in the Event, Fictional Character and Other categories. In fact, the top category in GraphQuestion is Other, which includes professions and common nouns.

A successful system for entity linking on question data needs to recognize and to link the whole variety of entity categories. We take this as the main requirement and consider the performance across all entity categories as the primary evaluation criterion for the entity linking models in this chapter. With our new architecture, we aim to show that comprehensive modeling of different contextual information will result in a better generalization and performance across various entity categories.

3.2.3 Approaches to Entity Linking

Previous machine learning approaches to entity linking usually rely on an off-the-shelf named entity recognizer to extract entity mentions in the input (Bunescu and Pasca, 2006; Cucerzan, 2007; Han and Sun, 2012; Francis-Landau et al., 2016). As a consequence, such approaches cannot handle entity categories that are not supplied by the named entity recognizer and they suffer from error propagation when the recognizer misclassifies a mention. To mitigate the error propagation,

Cucerzan (2012) has introduced the idea to perform mention detection and entity linking jointly using a linear combination of manually defined features. Luo et al. (2015) suggested a probabilistic graphical model for the joint prediction. Joint mention detection and disambiguation is essential for linking entities in questions. Consider the example "who does maggie grace play in taken?". At the mention detection stage, it is hard to determine if the token 'taken' is an entity mention or a common verb. It helps to consult a knowledge base and to use the information that 'taken' can refer to the movie 'Taken'. This is also a part of the entity disambiguation step, and it is therefore beneficial to model the two steps together. We follow the same concept in our approach for entity linking for question answering.

We rely on neural networks to model the context of the entities in the input text and in the knowledge base. Neural networks are much better in adapting to noise than the traditional feature-based models. Sun et al. (2015) were among the first to use neural networks to embed an entity mention and disambiguation candidates into fixed-size vectors. The vectors were used to determine, which of the candidate entities in the knowledge base is the most similar to the mention vector. Francis-Landau et al. (2016) have employed convolutional neural networks to extract features from the document context and mixed them with manually defined features. Sil et al. (2018) continued the work in this direction recently and applied convolutional neural networks to cross-lingual entity linking. Having proven themselves effective for the entity linking task, convolutional neural networks are also the corner stone of our entity linking architecture.

Approaches for entity linking on Twitter data as a form of short and noisy input represent the most relevant previous work for our goal of entity linking for question answering and fact verification. Guo et al. (2013b) have created the first entity linking dataset of around 1500 tweets and suggested a Structured SVM approach that handled mention detection and entity disambiguation together. Chang et al. (2014) describe the winning system of the NEEL 2014 competition on entity linking for Twitter: the system adapts a joint approach similar to Guo et al. (2013b), but uses the MART gradient boosting algorithm instead of the SVM and extends the feature set. The state-of-the-art system for entity linking on noisy data at the time of writing is S-MART (Yang and Chang, 2015), which extends the approach from Chang et al. (2014) by adding structured predictions that model non-overlapping entity mentions. The same team has subsequently applied S-MART to extract entities for a question answering system (Yih et al., 2015).

The described entity linking systems for Twitter data are limited in their application as stand-alone tools. The output of the S-MART system is available on some of the popular datasets, but not the system itself. Consequently, modern question answering approaches had to rely on off-the-shelf entity linkers that were designed for other domains. Reddy et al. (2016) have employed the Freebase online API in their question answering experiments that was since deprecated. A number of question answering systems have relied on DBpedia Spotlight to extract

entities (Lopez et al., 2016; Chen et al., 2016). DBpedia Spotlight (Mendes et al., 2011) uses document similarity vectors and manually defined features such as entity frequency. It was developed for entity linking in documents and evaluated on news data.

3.2.4 Variable Context Granularity Network

We are addressing the lack of a entity linking system for question answering with our work. In particular, we present a neural network architecture that focuses on short and noisy data and improves the performance across different entity categories against the previously available systems. We evaluate our entity linking system on question answering data and it is freely available as a part of the interactive demo application that we describe in Chapter 5.

In the previous section, we have already discussed that there are previous approaches, which have considered to make a joint mention detection and disambiguation decision and have shown that it improves the final entity linking result. We adopt this paradigm in our entity linking architecture, as well.

To implement this paradigm, we reduce the mention detection step to a mere unsupervised extraction of overlapping token n-grams that could be entity mentions, we refer to them as mention candidates. By default, it is reasonable to assume that any n-gram can be an entity mention (for instance, any token n-gram can be a movie or a song title).[7] If the mention detection step produces only candidates, the entity disambiguation step has to consider the optimal global assignment for the overlapping spans. Conceptually, the entity mention span that best matches an external referent is to be selected over the competing overlapping candidates. Consider another example from question answering: "Who is the author of Harry Potter and the Philosopher's Stone?". Without consulting external knowledge first, one would assume that many overlapping n-grams in this sentence can refer to an entity. Both 'Harry Potter and the Philosopher's Stone', 'Harry Potter' and 'The Philosopher's Stone' can refer to separate books, characters and franchises. Even 'The author' can refer to a movie or a book title. We would extract all these n-grams and forward them to the entity disambiguation step, where after consulting the knowledge base it is possible to conclude, that there indeed exists an entity 'Harry Potter and the Philosopher's Stone' that matches the longest span and the surrounding context (i.e. it is a book and one can ask who is an author of a book). We take that each token can only refer to a single entity at a time and other overlapping candidates can thus be ignored.

Deferring the final decision on the candidate mentions and the global assignment to the entity disambiguation step results in a single statistical model for the task. In this case, the initial extraction of mention candidates can be performed with a set of rules, since it only needs to limit the space of possible mentions.

[7] A curious reader may inspect the list of the shortest movie titles, many of which have a corresponding Wikidata entry: `https://www.imdb.com/list/ls064446756/`.

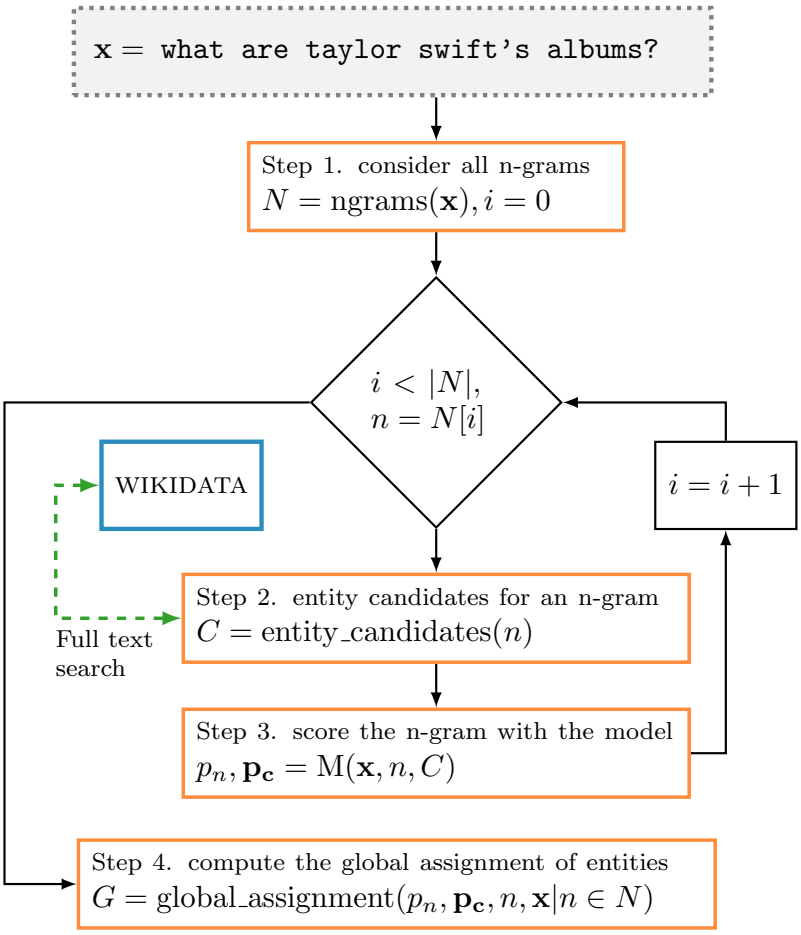

Figure 3.14: An overview of the processing steps in the entity linking system. The process extracts all token n-grams up to a fixed length in Step 1 and iterates over them. It extracts linking candidates from Wikidata for an n-gram (Step 2) and scores them with the disambiguation model (Step 3). All scored candidates are aggregated for the global assignment decision (Step 4) that ensures that each token in the input is linked only once.

Mention detection

The overall architecture of our entity linking system is depicted in Figure 3.14. We consider any n-gram in the input as a potential entity mention and do not rely on pre-existing NER tools. Thus, we limit the mention detection stage to simple pre-filtering of the extracted n-grams. We remove the n-grams that do not refer to any possible entity in the knowledge base using a simple string look up. Such setup allows us to apply our system on datasets with diverse entity categories, because we are no longer limited by the named entity categories recognized by an external NER tool.

From the input question $\mathbf{x}$, we extract all possible token n-grams N up to a certain length as entity mention candidates (Step 1). For each n-gram n, we look it up in the knowledge base using a full text search over entity labels (Step 2). That ensures that we find all entities that contain the given n-gram in the label. For

example for a unigram 'obama', we retrieve 'Barack Obama', 'Michelle Obama' etc. This step produces a set of entity disambiguation candidates C for the given n-gram n. We sort the retrieved candidates by length and cut off after the first 1000. That ensures that the top candidates in the list would be those that exactly match the target n-gram n.

In the next step, the list of n-grams N and the corresponding list of entity disambiguation candidates are sent to the entity linking model (Step 3). The model jointly performs the detection of correct mentions and the disambiguation of entities. We use the scores produced by the model in Step 3 directly to compute the global assignment of entities to the mentions in the input sentences (Step 4). The global assignment combines the scores into a probability of the whole sequence and ensures that no two overlapping mention candidates are selected.

Entity disambiguation

We design an end-to-end neural network to jointly predict an entity mention detection score for each n-gram and a ranking score for each entity disambiguation candidate for that mention. We hypothesize that aggregating information from different levels of context granularity (token and character level in the input text, entity label and relational level in the knowledge base) is essential for joint entity detection and disambiguation. The approaches that we have surveyed in Section 3.2.3 suggest features derived from different contexts, but do not aggregate them into a single end-to-end mention detection and linking system.

We present the neural Variable Context Granularity (VCG) architecture that aggregates and mixes contexts of different granularities to perform a joint mention detection and entity disambiguation. Figure 3.15 shows the layout of the network and its main parts as well as the context granularity levels. The input to the model is a list of question tokens $\mathbf{x}$, a token n-gram n and a list of candidate entities C. Then the model is a function $\mathrm{M}(\mathbf{x}, n, C)$ that produces a mention detection score p_n for the n-gram n and a ranking score p_c for each of the candidates $c \in C$: $p_n, \mathbf{p_c} = \mathrm{M}(\mathbf{x}, n, C)$, where $\mathbf{p_c}$ is a vector of size of C.

Dilated Convolutions To process sequential input, we use Dilated Convolutional Neural Networkss (DCNNs). Strubell et al. (2017) have recently shown that DCNNs are faster and as effective as recurrent models on the task of NER. We do not directly compare DCNNs against other sequence processing architectures (e.g. convolutional vs. recurrent), as this would be beyond the scope of this thesis. We define two modules: $\mathbf{DCNN}_w$ and $\mathbf{DCNN}_c$ for processing token-level and character-level input respectively. Here, both modules consist of a series of convolutions applied with an increasing dilation, as described in Strubell et al. (2017). The output of the convolutions is averaged and transformed by a fully-connected layer. We will re-use the dilated convolutions architecture for sequential input processing again in the subsequent Chapter 4.

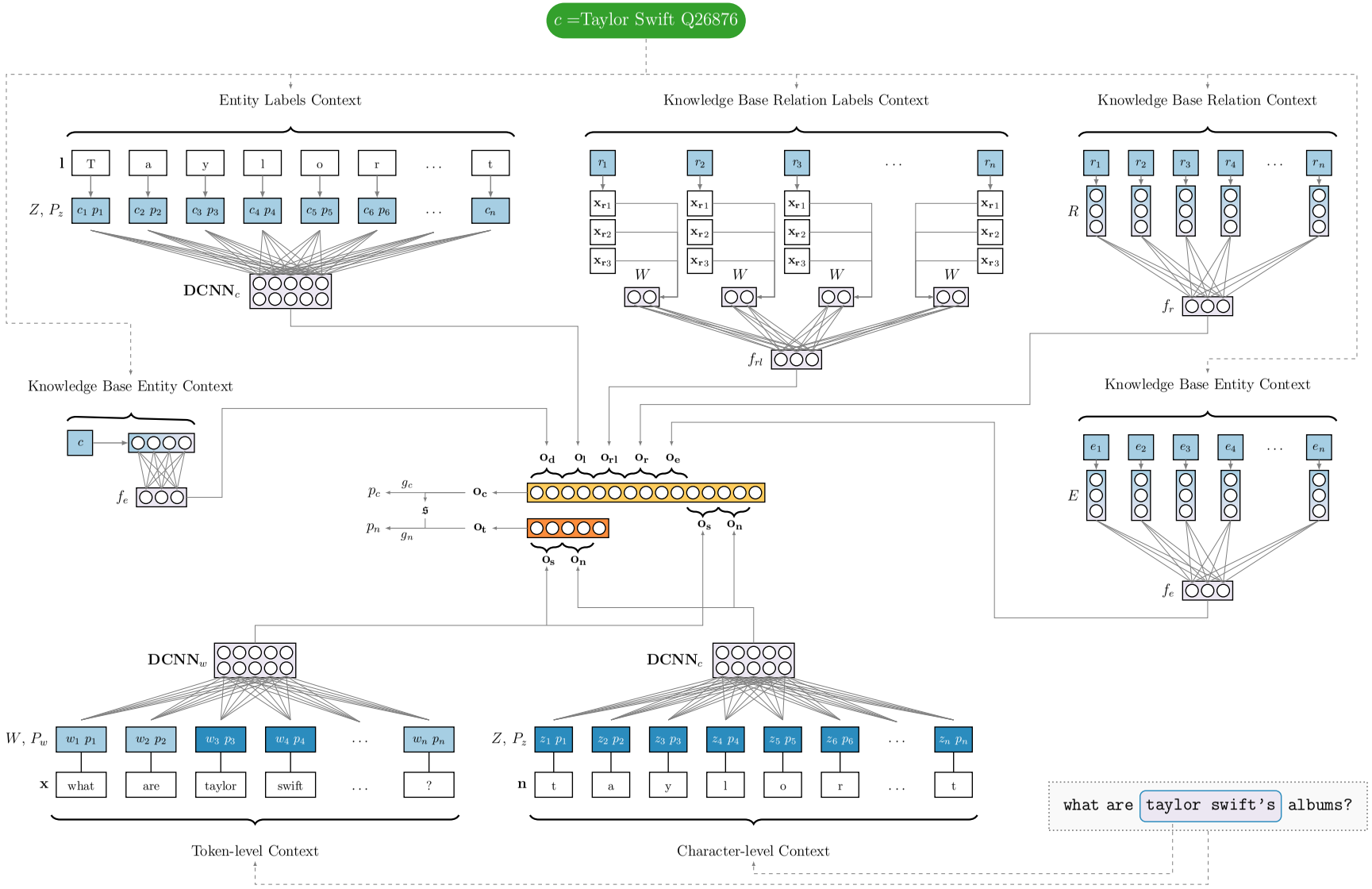

Figure 3.15: The architecture of the VCG network for a *single* n-gram and an entity candidate. The output vectors ($\mathbf{o_c}$, $\mathbf{o_t}$) (yellow and orange vectors in the very center) are aggregated over *all* n-grams for the global assignment.

Context Granularity Levels The *token level* encodes a list of input tokens $\mathbf{x}$ into a fixed size vector. We map the tokens in $\mathbf{x}$ to d_w-dimensional pre-trained word embeddings, using a matrix $\mathbf{W} \in \mathbb{R}^{|V_w| \times d_w}$, where $|V_w|$ is the size of the vocabulary. We again use 50-dimensional GloVe embeddings pre-trained on a 6 billion tokens corpus (Pennington et al., 2014). The word embeddings are concatenated with d_p-dimensional position embeddings $\mathbf{P_w} \in \mathbb{R}^{3 \times d_p}$. The position embedding marks each token as either a target n-gram token, any other token in the original input or the padding token. The concatenated embeddings are processed by $\mathbf{DCNN}_w$ to get a vector $\mathbf{o_s}$.

At the *character level*, we process the target token n-gram n on the basis of individual characters. We add one token on the left and on the right to the target mention to compensate for possible mention border detection errors and map the string of characters to d_z-character embeddings, $\mathbf{Z} \in \mathbb{R}^{|V_z| \times d_z}$, where $|V_z|$ is the size of the character set. We concatenate the character embeddings with d_p-dimensional position embeddings $\mathbf{P_z} \in \mathbb{R}^{|n| \times d_p}$ and process them with $\mathbf{DCNN}_c$ to get a feature vector $\mathbf{o_n}$. We use $\mathbf{DCNN}_c$ with the same learned parameters to encode the label of a candidate entity from the knowledge base as a vector $\mathbf{o_l}$. The parameter sharing between mention encoding and entity label encoding ensures that the representation of a mention is similar to the entity label on the character level.

At the *knowledge base structure level*, we model the entities and relations that are connected to the candidate entity c. First, we map a list of relations $\mathbf{r}$ of the candidate entity to d_r-dimensional pre-trained relation embeddings, using a matrix $\mathbf{R} \in \mathbb{R}^{|V_r| \times d_r}$, where $|V_r|$ is the number of relation types in the knowledge base. We transform the relation embeddings with a single fully-connected layer f_r and then apply a max pooling operation to get a single relation vector $\mathbf{o_r}$ per entity. Similarly, we map all entities that are immediately connected with the candidate entity $\mathbf{e}$ to d_e-dimensional pre-trained entity embeddings, using a matrix $\mathbf{E} \in \mathbb{R}^{|V_e| \times d_e}$, where $|V_e|$ is the number of entities in the knowledge base. The entity embeddings are transformed by a fully-connected layer f_e and then also pooled to produce the output $\mathbf{o_e}$. The embedding of the candidate entity itself is also transformed with f_e and is stored as $\mathbf{o_d}$. To train the knowledge base embeddings, we use the TransE algorithm of Bordes et al. (2013).

Finally, the *knowledge base lexical level* is similar to the knowledge base structure level, but instead of embeddings learned from the knowledge base structure, we take the labels of the relations in $\mathbf{r}$ to compute lexical relation embeddings. For each $r \in \mathbf{r}$, we tokenize the label and map the tokens $\mathbf{x_r}$ to word embeddings, using the word embedding matrix $\mathbf{W}$. To get a single lexical embedding per relation, we apply max pooling and transform the output with a fully-connected layer f_{rl}. The lexical relation embeddings for all relations of the candidate entity are pooled into a single vector $\mathbf{o_{rl}}$.

Context Aggregation The information from the different granularity levels of context is aggregated and is transformed by a sequence of fully-connected layers

into a final vector $\mathbf{o_c}$ for the n-gram n and the candidate entity c (see the center of Figure 3.15). The vectors for each candidate are aggregated into a matrix $O = [\mathbf{o_c}|c \in C]$.

To get the ranking score p_c for each entity candidate c, we apply a single fully-connected layer g_c on the concatenation of $\mathbf{o_c}$ and the summary vector $\mathfrak{s}$: $p_c = g_c(\mathbf{o_c}\|\mathfrak{s})$. We apply element-wise max pooling on the matrix of candidate embeddings O to get a single summary vector $\mathfrak{s}$ for all entity candidates for n. The rationale is to compute the score for each disambiguation candidate in the context of all the other available candidates.

To compute the mention detection score for the n-gram n, we separately concatenate the vectors for the token context $\mathbf{o_s}$ and the character context $\mathbf{o_n}$ and transform them with an array of fully-connected layers into a vector $\mathbf{o_t}$. We concatenate $\mathbf{o_t}$ with the summary vector $\mathfrak{s}$ to add information about disambiguation candidates C and apply another fully-connected layer to get the mention detection score $p_n = \sigma(g_n(\mathbf{o_t}\|\mathfrak{s}))$. It is clear from Figure 3.15 that the summary vector already contains the information from the token-level and the character-level context. The token and character features describe the entity mention in the text and are thus central for learning a representation for the mention. It has been often shown that additional connections in the network from the input to the learned representations that are informed by the task rationale are beneficial (cf. the original motivation for the attention mechanism in NLP in Rocktäschel et al., 2016).

Global Entity Assignment

The first step in our system is extracting all possible overlapping n-grams from an input text. We assume that each token in the input text can only refer to a single entity and therefore we need to resolve overlaps by computing a global assignment using the model scores for each n-gram (Step 4 in Figure 3.14). If the mention detection score p_n is above the 0.5-threshold, the n-gram is predicted to be a correct entity mention and the ranking scores $\mathbf{p_c}$ are used to disambiguate it to a single entity candidate. N-grams that have p_n lower than the threshold are filtered out.

We follow Guo et al. (2013a) in computing the global assignment and hence, arrange all n-grams selected as mentions into non-overlapping combinations and use the individual scores p_n to compute the probability of each combination. The combination with the highest probability is selected as the final set of entity mentions. In practice, we have observed the same effect as described by Strubell et al. (2017), namely that DCNNs are able to capture dependencies between different entity mentions in the same context and do not tend to produce many overlapping mentions.

Composite Loss Function

Our model computes the two scores for each n-gram: the mention detection score p_n and the disambiguation score p_c. We optimize the parameters of the whole model jointly and use a loss function that combines penalties for the both scores for all n-grams in the input question:

$$\mathcal{L} = \sum_{n \in N} \sum_{c \in C_n} \mathcal{M}(t_n, p_n) + t_n \mathcal{D}(t_c, p_c), \tag{3.8}$$

where t_n is the target for mention detection and is either 0 or 1, t_c is the target for disambiguation and ranges from 0 to the number of candidates $|C|$.

For the mention detection loss $\mathcal{M}$, we include a weighting parameter α for the negative class as the majority of the instances in the data are negative:

$$\mathcal{M}(t_n, p_n) = -t_n \log p_n - \alpha(1 - t_n) \log(1 - p_n) \tag{3.9}$$

The disambiguation detection loss $\mathcal{D}$ is a maximum margin loss:

$$\mathcal{D}(t_c, p_c) = \frac{\sum_{i=0}^{|C|} \max(0, (m - p_c[t_c] + p_c[i]))}{|C|}, \tag{3.10}$$

where m is the margin value. We set $m = 0.5$, whereas the α weight is optimized with the other hyper-parameters.

Architecture Comparison

Our model architecture follows some of the ideas presented in Francis-Landau et al. (2016): they suggest computing a similarity score between an entity and its context for different context granularity levels. Francis-Landau et al. (2016) experiment on entity linking for Wikipedia and news articles and consider the word-level and document-level contexts for entity disambiguation. As described above, we also incorporate different levels of context granularity with a number of key differences:

(1) we operate on the token and character level, thus including a more fine-grained range of contexts;

(2) the knowledge base contexts that Francis-Landau et al. (2016) use are the Wikipedia title and the article texts — we, on the other hand, employ the structure of the knowledge base and encode relations and related entities;

(3) Francis-Landau et al. (2016) separately compute similarities for each type of context, whereas we mix them in a single end-to-end architecture;

(4) we do not rely on manually defined features in our model and propose a neural network architecture instead;

emb. size					filter size		
d_w	d_z	d_e	d_r	d_p	$\mathbf{DCNN}_w$	$\mathbf{DCNN}_c$	α
50	25	50	50	5	64	64	0.5

Table 3.5: Best hyper-parameter configuration for the VCG model. The parameters were selected after a random search on the WebQSP development set.

	All entities		
	P	R	F1
Heuristic baseline	0.286	0.621	0.392
Simplified VCG	0.804	**0.654**	0.721
VCG	**0.823**	0.646	**0.724**

Table 3.6: Evaluation results on the development set of WebQSP entity linking dataset.

The aforelisted improvements that we introduce with our model are motivated by the downstream tasks of fact verification and foremost question answering. With the flexible VCG model that covers the broadest set of contexts we aim to improve the entity linking for noisy question answering data.

Model training

The hyper-parameters of the model, such as the dimensionality of the layers and the size of embeddings, are optimized with random search on the WebQSP development set. The model was particularly sensitive to tuning of the negative class weight α (see Equation 3.8). Table 3.5 lists the main selected hyper-parameters for the VCG model[8] and we also report the results for each model's best configuration on the development set in Table 3.6.

3.2.5 Datasets and Evaluation Methodologies

We compile two new datasets for entity linking on question answering data to evaluate the proposed models in the challenging short and noisy text scenario. We derive the datasets from publicly available question answering data: WebQSP (Yih et al., 2016) and GraphQuestions (Su et al., 2016).

WebQSP contains questions that were originally collected for the WebQuestions dataset from web search logs (Berant et al., 2013). They were manually annotated with SPARQL queries that can be executed against a knowledge base to retrieve the correct answer to each question. Additionally, the annotators have also selected the main entity in the question that is central to finding the answer. The

[8] The complete list of hyper-parameters and model characteristics can be found in the code repository: `https://github.com/UKPLab/starsem2018-entity-linking`

	#Questions	#Entities
WebQSP Train	3098	3794
WebQSP Test	1639	2002
GraphQuestions Test	2182	2336

Table 3.7: Statistics for the entity annotated question answering datasets.

annotations and the query use identifiers from the Freebase knowledge base.

We extract all entity identifiers from the SPARQL query (these are the entities that are mentioned in the question). For the main entity, we also store the corresponding span in the text, as annotated in the dataset. To use Wikidata in our experiments, we translate the Freebase identifiers to Wikidata IDs using the mapping information available in Wikidata (for Wikidata entities, the corresponding Freebase IDs are manually added by the contributors).

The second dataset, GraphQuestions, was created by collecting manual paraphrases for automatically generated questions (Su et al., 2016). The dataset is meant to test the ability of the system to understand different wordings of the same question. In particular, the paraphrases include various references to the same entity, which creates a challenging task for an entity linking system. The following are three example questions from the GraphQuestions dataset that contain a mention of the same entity (marked in bold):

1. what is the rank of marvel's **iron man**?
2. **iron-man** has held what ranks?
3. **tony stark** has held what ranks?

GraphQuestions does not contain main entity annotations, but includes a SPARQL query structurally encoded in JSON format. The queries were constructed manually by identifying the entities in the question and selecting the relevant knowledge base relations. We extract gold entities for each question from the SPARQL query and map them to Wikidata.

We split the WebQSP training set into train and development subsets to optimize the neural model. We use GraphQuestions only in the evaluation phase to test the generalization power of our model. The sizes of the constructed datasets in terms of the number of questions and the number of entities are reported in Table 3.7. In both datasets, we only keep questions that contain at least one entity mention.

We use precision, recall and F1 scores to evaluate and compare the approaches. We follow Carmel et al. (2014) and Yang and Chang (2015) and define the scores on a per-entity basis. Since there are no mention boundaries for the gold entities, an extracted entity is considered correct if it is present in the set of the gold entities

for the given question. This is a departure from the standard entity linking metrics that also take into account the overlap between the detected entity mention and the gold mention token span. However, for the downstream applications that we have in mind it is sufficient to correctly disambiguate the entity and the exact mention span is not required. Consider the question answering example in Figure 3.11 and the fact checking example in Figure 3.12, where we first perform entity linking. In both cases, we need only the set of disambiguated entities mentioned in the input, but not their exact token spans.

We define precision (P) and recall (R) based on gold and predicted entity sets for each input $\mathbf{x}$ in the test set D. We compute the metrics in the micro and macro setting. In the micro setting we compute precision and recall over all gold entities and model predictions:

$$\mathrm{P} = \frac{\sum_{\mathbf{x} \in D} |\{\text{gold e in } \mathbf{x}\} \cup \{\text{predicted e in } \mathbf{x}\}|}{\sum_{\mathbf{x} \in D} |\{\text{predicted e in } \mathbf{x}\}|} \tag{3.11}$$

$$\mathrm{R} = \frac{\sum_{\mathbf{x} \in D} |\{\text{gold e in } \mathbf{x}\} \cup \{\text{predicted e in } \mathbf{x}\}|}{\sum_{\mathbf{x} \in D} |\{\text{gold e in } \mathbf{x}\}|}, \tag{3.12}$$

$$\tag{3.13}$$

where e stays for entities. The macro values are computed by breaking down the test data by entity category first and averaging the precision and recall values afterwards:

$$\mathtt{mP} = \sum_{k \in K} \left(\frac{\sum_{\mathbf{x} \in D} |\{\text{gold e in } \mathbf{x}\} \cup \{\text{predicted e in } \mathbf{x}\} \cup \{\text{e in } k\}|}{\sum_{\mathbf{x} \in D} |\{\text{predicted e in } \mathbf{x}\} \cup \{\text{e in } k\}|} \right) / |K| \tag{3.14}$$

$$\mathtt{mR} = \sum_{k \in K} \left(\frac{\sum_{\mathbf{x} \in D} |\{\text{gold e in } \mathbf{x}\} \cup \{\text{predicted e in } \mathbf{x}\} \cup \{\text{e in } k\}|}{\sum_{\mathbf{x} \in D} |\{\text{gold e in } \mathbf{x}\} \cup \{\text{e in } k\}|} \right) / |K|, \tag{3.15}$$

$$\tag{3.16}$$

where e stays for entities and K is the set of entity categories in D. The F-score is always computed as a harmonic mean of precision and recall. The micro evaluation describes how good the system performs on average, while the macro scores offer a more fair comparison towards the small entity categories. We have argued in the beginning of this section that a steady performance on all entity categories is crucial for the downstream applications of entity linking.

For the WebQSP dataset, we additionally perform a separate evaluation using only the information on the main entity. The main entity has the information on the boundary offsets of the correct mentions and therefore for this type of evaluation, we consider the extracted entity correct only if it has a token overlap with the correct entity mention. Typical question answering systems need at least one entity per question to attempt finding the correct answer. Thus, evaluating using the main entity shows how the entity linking system fulfills this minimum requirement for the downstream application.

	All entities					
	P	R	F1	mP	mR	mF1
DBpedia Spotlight	0.705	0.514	0.595	0.572	0.392	0.452
S-MART	0.666	**0.772**	0.715	0.607	**0.610**	0.551
Heuristics baseline	0.302	0.608	0.404	0.330	0.537	0.378
Simplified VCG	**0.837**	0.621	0.713	0.659	0.494	0.546
VCG	0.826	0.653	**0.730**	**0.676**	0.519	**0.568**

Table 3.8: Entity linking evaluation results on the WebQSP test set for all question entities. The m prefix stands for *macro* results, which are computed per entity category and then averaged.

3.2.6 Experimental Results

Existing Systems In our experiments, we compare our proposed architecture to DBpedia Spotlight that was used in several question answering systems and represents a strong baseline for entity linking.[9] In addition, we are able to compare to the state-of-the-art S-MART system, since its output on the WebQSP dataset was publicly released.[10] We also include a heuristics baseline that ranks candidate entities according to their frequency in Wikipedia. This baseline represents a reasonable lower bound for a Wikidata based approach.

Simplified VCG To test the effect of the end-to-end context encoders of the VCG network, we define a model that uses a set of features commonly used for entity linking on noisy data. In particular, we employ features that cover (1) frequency of the entity in Wikipedia, (2) edit distance between the label of the entity and the token n-gram, (3) number of entities and relations immediately connected to the entity in the knowledge base, (4) word overlap between the input question and the labels of the connected entities and relations, (5) length of the n-gram. We also add an average of the word embeddings of the question tokens and, separately, an average of the embeddings of tokens of entities and relations connected to the entity candidate. We train the simplified VCG model by optimizing the loss function in Equation 3.8 on WebQSP the same way as for VCG.

Results Table 3.8 lists results for the heuristics baseline, for the simplified VCG baseline and for the proposed VCG model on the test set of WebQSP. The simplified VCG model outperforms DBpedia Spotlight (0.713 F1 vs. 0.595 F1) and achieves a result close to the S-MART model (0.713 F1 vs. 0.715 F1). We observe that our VCG model has the best F1 among the tested models and achieves the most gains in precision compared to S-MART, the previous state-of-the-art model for question answering data (0.826 P for VCG vs. 0.666 P for S-MART). The recall

[9] We use the online end-point: `http://www.dbpedia-spotlight.org/api`

[10] The output on WebQSP produced by Yih et al. (2015) can be found at `https://github.com/scottyih/STAGG`. The S-MART system itself is not openly available.

	Main entity		
	P	R	F1
DBpedia Spotlight	0.668	0.595	0.629
S-MART	0.634	**0.899**	0.744
Heuristics baseline	0.282	0.694	0.401
Simplified VCG	**0.804**	0.728	0.764
VCG	0.793	0.766	**0.780**

Table 3.9: Entity linking evaluation results on the WebQSP test set for the main entity in the question.

	P	R	F1
DBpedia Spotlight	0.386	0.453	0.417
Heuristics baseline	0.113	**0.509**	0.184
Simplified VCG	0.579	0.374	**0.454**
VCG	**0.589**	0.354	0.442

Table 3.10: Entity linking evaluation results on GraphQuestions dataset for all question entities.

of the VCG model is worse than of S-MART, but VCG achieves a better balance between the precision and recall and delivers the best F-score across all setups. Considering only the main entity in Table 3.9, we observe the same trends. The simplified VCG baseline outperforms both DBpedia Spotlight and S-MART in the main entity evaluation and the VCG model produces the best result overall.

VCG outperforms the simplified VCG baseline that was trained by optimizing the same loss function but uses manually defined features. Thereby, we confirm the advantage of the mixing context granularities strategy that was proposed in our work. Most importantly, the VCG model achieves the best macro result (mF1 in Table 3.9), which indicates that the model has a consistent performance on different entity categories.

We further evaluate the developed VCG architecture on the GraphQuestions dataset against the DBpedia Spotlight. We use this dataset to evaluate VCG in an out-of-domain setting: neither our system nor DBpedia Spotlight were trained on it. The results for each model are presented in Table 3.10. We can see from the heuristics baseline results, which provides a lower bound for performance, that GraphQuestions is a much more difficult benchmark for entity linking (cf. 0.184 F1 for Heuristics baseline in Table 3.10 with 0.404 F1 for Heuristics baseline in Table 3.8). This agrees with the generally much lower question answering results on this dataset compared to other question answering benchmarks. The VCG model shows the overall F-score result that is better than the DBpedia Spotlight baseline by a wide margin. It is notable that VCG achieves higher precision val-

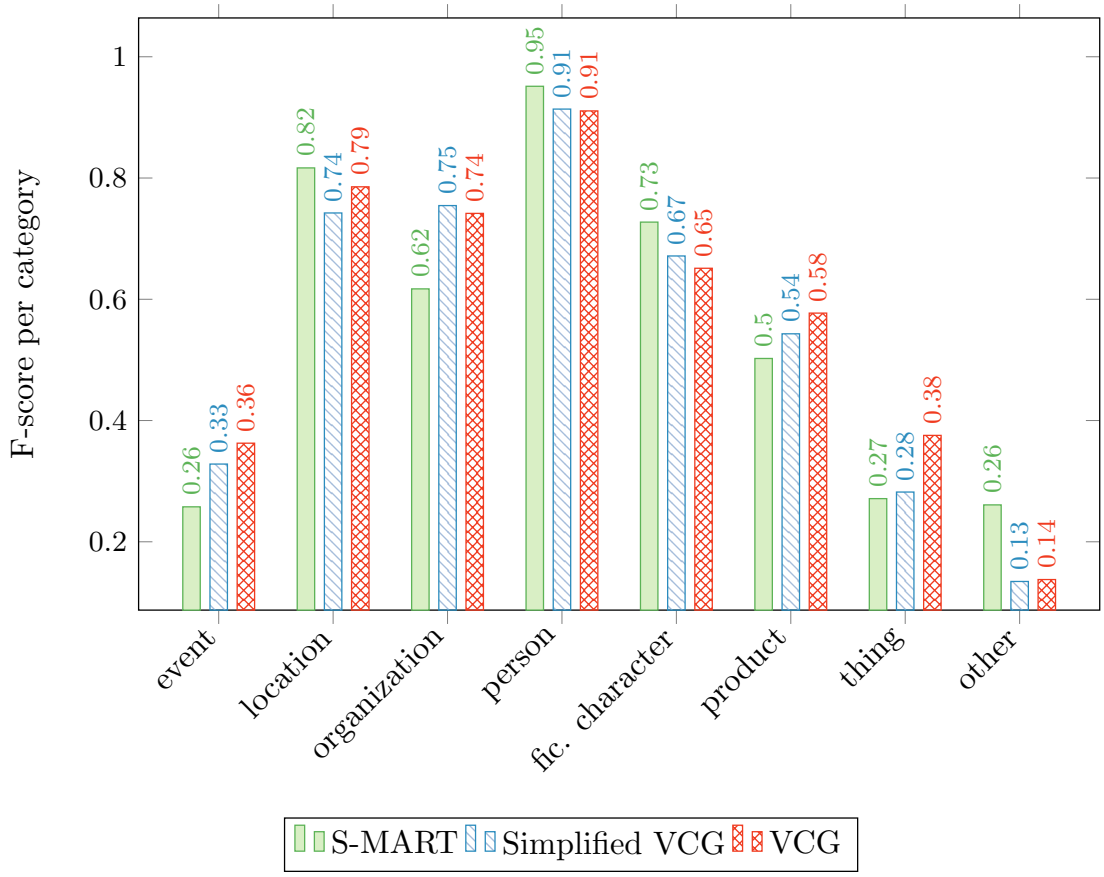

Figure 3.16: Performance across entity categories on the WebQSP test set for the VCG architecture and the baselines.

ues as compared to other approaches and manages to keep a satisfactory level of recall. At the same time, the simplified VCG delivers a slightly better result on GraphQuestions, without achieving either the best precision or recall among the tested models.

3.2.7 Error Analysis

To understand the performance difference between the approaches and the gains of the VCG model, we analyze the results per entity category on the WebQSP test set (see Figure 3.16). We compare to the simplified feature-based version and to the S-MART system. We see that the S-MART system is slightly better in the disambiguation of Locations, Person names and a similar category of Fictional Character names. For instance, S-MART correctly recognizes 'robin hood' in "who plays robin hood in prince of thieves?", which is categorized as a Fictional Character name. S-MART also has a considerable advantage in processing entities in the Other category, which includes professions and other common nouns, such as 'head judge' in "what is the head judge of the supreme court called?".

Our VCG approach has an edge in such entity categories as Organization, Product and Thing. The Product category includes movies, book titles and songs, which

	All entities					
	P	R	F1	mP	mR	mF1
VCG	**0.826**	**0.653**	**0.730**	**0.676**	**0.519**	**0.568**
– token level	0.812	0.618	0.702	0.664	0.474	0.530
– character level	0.820	0.573	0.675	0.667	0.404	0.471
– knowledge base structure level	0.728	0.576	0.643	0.549	0.427	0.461
– knowledge base lexical level	0.807	0.617	0.699	0.643	0.454	0.508

Table 3.11: Ablation experiments for the VCG model on the WebQSP entity linking dataset. The m prefix stands for *macro*. VCG is the full entity linking model proposed in our work and the rest is the same model but without the specified context granularity level.

are particularly hard to identify and disambiguate since any sequence of words can be a title. For instance, VCG recognizes the movie title name 'return of the jedi' in "who played darth vader at the end of return of the jedi?". VCG is also considerably better in recognizing Events, as 'the 2000 fa cup final' in "who won the 2000 fa cup final?". The simplified VCG is behind the VCG model on most categories except Organization and Fictional Character where it slightly outperforms the full VCG model.

We can conclude that the future development of the VCG architecture should focus on the improved identification and disambiguation of professions. The Other category, which includes professions, is overall the most challenging category for the evaluated models. In Figure 3.16, the best result for this category (0.26 for S-MART) is lower then the model results across all other categories.

To analyze the effect of mixing various context granularities on the model performance, we include ablation experiment results for the VCG model on the WebQSP dataset in Table 3.11. We report results for our VCG architecture and for its variants without the individual context granularity levels, leaving them out one at a time (see Section 3.2.4 for the description of granularity levels). We use the same metrics as in the main evaluation.

The removal of the knowledge base structure information encoded in entity and relation embeddings results in the biggest performance drop of 9 percentage points for F-score (cf. 0.730 F1 for VCG and 0.643 F1 for '– knowledge base structure context' in Table 3.11). We see a big drop in performance for '– knowledge base structure context' with the macro metrics as well. The character-level information also proves to be highly important for the final state-of-the-art performance (cf. 0.730 F1 for VCG and 0.675 F1 for '– character context' in Table 3.11). These two aspects of the VCG model, the comprehensive representation of the knowledge base structure and the character-level information, are two of the main differences of our approach to the previous work, as discussed in Section 3.2.4. Finally, we see that excluding the token context and the lexical information about the related

knowledge base relations also decrease the results across all metrics, albeit less dramatically.

In the second part of this chapter (Section 3.2), we have discussed entity linking with Wikidata. Alongside relation extraction, entity linking is the second of the two main fundamental blocks of grounded semantic parsing (see again Figure 2.11 in the previous chapter). We have narrowed down the task description for entity linking on noisy question answering data and have contributed a new entity linking architecture. We introduced the variable context granularities model (VCG) that aggregates information from the token level and character level in the text and the lexical and relational level in the knowledge base. VCG achieves state-of-the-art results for linking entities on two datasets: WebQSP (see Table 3.8) and GraphQuestions (see Table 3.10). In particular, it outperforms the existing S-MART system, which was used for entity liking in the question answering task before (Yih et al., 2015). VCG is freely available as an open source repository[11] and as a part of the interactive demo application that we describe in Chapter 5. In the next section, we use our entity linking model to enhance NLP applications with world knowledge from Wikidata by linking the entities in the input text.

3.3 Enhancing Applications with World Knowledge

The linking methods presented in this chapter represent the basic building blocks that allow us to connect a text to the structure of the external knowledge base. The entity linking system identifies and disambiguates mentions of the knowledge base entities in a text, and the relation extraction system classifies relations between the identified entities into one of the knowledge base relation types. Information on the knowledge base units expressed in the text allows us to access more of the knowledge that is connected to those units and thus enhance the textual input with more data. We explore two collaboration projects that aimed to use additional external knowledge from Wikidata to enrich natural language understanding applications.

In previous studies, it has been shown that injecting various external information can lead to substantial improvements on semantic tasks. Weissenborn et al. (2018a) used the WordNet lexical database (Fellbaum, 1998) to add more information to a document question answering model of Weissenborn et al. (2018b). Mihaylov and Frank (2018) added the information from ConceptNet (a combination of lexical and commonsense information) to improve on the cloze-style text comprehension task. However, there are unclear dependencies between the nature of the task and the external information source that is being used. Not every external information is useful for every semantic understanding task. Glockner et al. (2018) show that simple lexical information, such as contained in WordNet, is not enough to overcome the difficulties of the natural language inference task (Bowman et al., 2015). This finding might have been already anticipated in the previous work:

[11] `https://github.com/UKPLab/starsem2018-entity-linking`

Zhai et al. (2016) have argued that information in WordNet overlaps with pre-trained word embeddings. The type of information and the coverage also play an important role.

Taken together, this suggests that linking the input to a large-scale knowledge base might enhance the approach with useful information which is not accessible via standard word embeddings or via other external lexical resources such as WordNet or ConceptNet. Nevertheless, it is subject to empirical verification and we make a step into this direction in the last part of this chapter.

Entity linking is the first step for linking the text to a knowledge base and thus is the basis for the introduction of external information into an NLP application. We focus on the application of our entity linker to two semantically-rich NLP tasks in this section: argument reasoning comprehension (in Section 3.3.1) and fact verification (in Section 3.3.2). The VCG entity linking model that was developed as a part of this thesis project has been empirically proven to be robust on the question answering data (Section 3.2.6). It covers a broad range of possible entity categories that are encoded in Wikidata and is well suited for heterogeneous data. We rely on our VCG entity linking architecture to enhance the textual input with external data in the both case studies below. These experiments serve as an extrinsic evaluation for our entity linking model and demonstrate that it is able to generalize to new data.

The publications that cover these case studies stem from collaborations where we provided the entity linking system. We summarize the approaches focusing on the entity linking, the external knowledge aspect of the system and its effect on the result. We refer the interested reader to the original papers for more details on other components. Our papers Botschen et al. (2018b)[12] and Hanselowski et al. (2018)[13] are foundational to this section.

3.3.1 Application: Argument Reasoning Comprehension

Common-sense argumentative reasoning is a task that entails deep semantic understanding of the argument structure and involves inference and logical reasoning. We consider the version of the argumentative reasoning problem that was formalized by Habernal et al. (2018a) for a challenge at SemEval-2018[14] as a binary warrant selection task. Their task has focused on reasoning about common knowledge and world events and entities. After reviewing the participating systems, Habernal et al. (2018b) hypothesize that external world knowledge may be essential for argument reasoning comprehension. For instance, to reason about the following two arguments from the Habernal et al. (2018a)'s dataset: "People

[12] My contribution in this paper is the following: application of the entity linking system on the task data, knowledge base embeddings, running the experiments and the result analysis with respect to the Wikidata information.

[13] My contribution in this paper is the following: annotation of the data with Wikidata entities, adjusting the entity linking approach, analysis with respect to the entities.

[14] SemEval-2018 Task 12: `https://competitions.codalab.org/competitions/17327`

can choose not to use Google." and "They can opt-out from being indexed by their search engine.", it is helpful to recognize the relation between the entities 'Google' and 'search engine'. This information can be extracted from Wikidata, after the entities in the input are linked.

Habernal et al. (2018a) define the task as selecting one of two possible warrants to fill a gap in an argumentative reasoning chain: given a debate title (a), reason (b) and claim (c), a system has to choose the correct warrant (i) or (ii) to use between the reason and the claim:

(a) Title: Can companies be trusted?

(b) Reason: Corporations have only one goal: to make a profit.

  (i) warrant: they do not have to satisfy customers to make a profit

  (ii) warrant: they have to satisfy customers to make a profit

(c) Claim: Companies can't be trusted.

The argumentation chains were constructed to require logical reasoning. The two warrant options were intentionally made very similar, but with opposing meanings.

In Botschen et al. (2018b), we explore the idea of using knowledge about prototypical events from FrameNet and fact knowledge about concrete entities from Wikidata to solve the argumentative warrant selection task. We find that both resources contribute to an improvement over the non-enriched approach and point out two persisting challenges: first, the integration of many annotations of the same type, and second, fusion of complementary annotations. However, we also conclude that the external knowledge might not be the main component necessary to achieve substantial improvements on the argumentative reasoning task, but that it is important to model the chain of reasoning with logic-based methods.

Entity- and Event-knowledge for Argumentative Reasoning

For the warrant selection task as described above, we do not pursue a semantic parsing approach or direct grounding in a knowledge base, but rather use the entity linking as a way to provide additional context with the input. In Botschen et al. (2018b), we have experimented with FrameNet as a second source of additional context. Wikidata entities add information about world entities, whereas we use FrameNet to annotate events in the same input text (Figure 3.17).

The Berkeley FrameNet (Baker et al., 1998; Ruppenhofer et al., 2016) is an ongoing project for manually building a large lexical-semantic resource with expert annotations. It embodies the theory of frame semantics (Fillmore, 1976): frames capture units of meaning corresponding to prototypical events (see also our overview in Section 2.2.1). Each frame can have a number of semantic roles, which describe the participants of the event described by the event. FrameNet consists

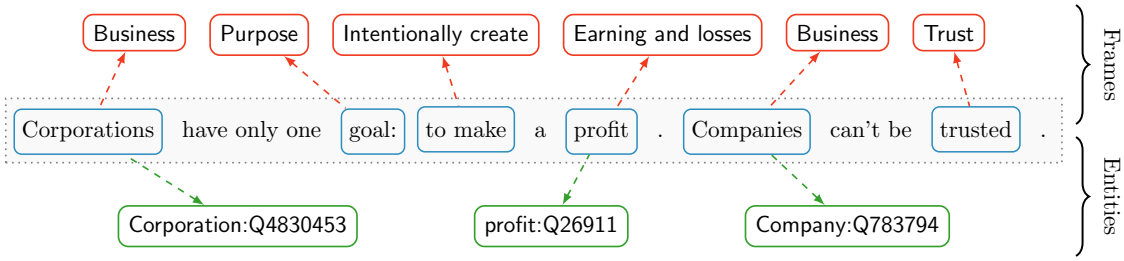

Figure 3.17: Overlapping Wikidata and FrameNet annotations for an example from the argument reasoning comprehension dataset.

of two parts, a lexicon that maps predicates to frames they can evoke, and fully annotated texts. For example, the verb 'buy' can evoke either the frame 'Commerce_sell' or 'Fall_for', depending on the context ('buying goods' versus 'buying a lie'). Furthermore, the lexicon gives access to frame-specific role-labels (e.g., 'Buyer', 'Goods' or 'Deception', 'Victim') as applied in semantic role labeling. We use FrameNet 1.5 which contains more than 1000 frames and more than 12,000 distinct predicate-frame combinations[15]. We use only frame annotations in the experiments in this section and do not include semantic roles. Our goal is to investigate the combination of entity and event information for argumentative reasoning.

Combining multiple levels of annotations has been shown to help in different NLP contexts. For instance, Khashabi et al. (2018) combine semantic role labeling, co-reference annotations and syntax in a text comprehension system. At the same time, we have noted above that not every extension with external additional information helps (Glockner et al., 2018). The two resources we chose, Wikidata and FrameNet, provide information beyond the simple lexical relations, such as those encoded in WordNet, and thus, have a potential to enhance the underlying model with other kind of external world knowledge. On the one hand, identifying frames unveils the event that is happening and that is described in the input. On the other hand, linking entities to a knowledge base disambiguates the entities that are mentioned in the context of the frames and taps into the vast information available in the knowledge base.

The recent work on event semantics hints at the entity and event annotations complementing each other. Additional information about entities benefits event prediction, as was shown by Ahrendt and Demberg (2016) and Botschen et al. (2018a). And conversely context information about events benefits the prediction of implicit arguments and entities (Cheng and Erk, 2018). The complementarity of the information in Wikidata and FrameNet is further affirmed by efforts on resource alignment. Wikidata properties, which are binary relations between entities, can be viewed as frames with only two participants. Thus, the correspondence in meaning between some frames in FrameNet and Wikidata properties

[15] `framenet.icsi.berkeley.edu/fndrupal`

were used to align the resources (Mousselly-Sergieh and Gurevych, 2016). The alignments allow to estimate the information overlap between the two knowledge resources. The best alignment approach of Mousselly-Sergieh and Gurevych (2016) only maps 37 % of the total Wikidata properties to FrameNet frames. This demonstrates that the most information in Wikidata and FrameNet does not overlap, but is rather complementary.

Our approach extends the Habernal et al. (2018a)'s model for argument warrant selection with two external knowledge schemata, FrameNet and Wikidata, to explore their effects. The intuition behind extending the model with FrameNet and Wikidata knowledge can be explained with the dataset example in Figure 3.17. The frame annotations describe six events in the example and disambiguate the tokens 'companies' and 'corporations' to the same frame 'Business'. The entity annotations include three distinct entities, which can be extended with further detailed information from Wikidata: for instance, Company:Q783794 is in fact a subtype of Corporation:Q4830453 in Wikidata and both entities are directly related to the entity profit:Q26911.

In the next section, we first examine the effect of both annotations separately and then we explore whether a joint annotation benefits from the inherent complementarity of the schemata in FrameNet and Wikidata and eventually leads to a better annotation coverage. We enhance the model, which was provided as a baseline for the argument reasoning comprehension task by Habernal et al. (2018a), and contrast three model configurations: +FrameNet, +Wikidata and +FrameNet/Wikidata. Our main contribution with this experiment is the proof of concept for enriching the argumentative reasoning with world information.

Evaluation

The baseline provided by Habernal et al. (2018a) is an intra-warrant attention model that reads in pre-trained *Word2Vec* vectors (Mikolov et al., 2013) of all words in the title, the claim and the reason and adapts attention weights for the decision between the warrant (i) and the warrant (ii).

We use our VCG entity linker for Wikidata that we have described in Section 3.2 and apply it directly on the claim, the reason and the alternative warrants (i, ii) to get the entity annotations. The entity linker is applied off-the-shelf and was not re-trained or fine-tuned on the argument comprehension data. For frame annotations, we use a freely available frame identification system from Botschen et al. (2018a). We employ pre-trained vector representations to encode information from FrameNet and Wikidata and add them into the model architecture, similarly as word embeddings are used to add distributional information. We use the pre-trained frame embeddings ($\mathbf{W_{frame}} \in \mathbb{R}^{N_{frames} \times 50}$) that are learned with *TransE* (Bordes et al., 2013) on the structure of the FrameNet hierarchy (Botschen et al., 2017). We also used *TransE* to pre-train entity embeddings ($\mathbf{W_{entity}} \in \mathbb{R}^{N_{entities} \times 100}$) on the Wikidata graph (the same embeddings that we have used in Section 3.2.4). The

Approach	Dev.	mean (±)	Test	(±)	max Test
Habernal et al. (2018a) (reimpl.)	0.6712	0.0155	0.5570	0.0184	0.5878
+Wikidata	0.6623	0.0071	**0.5680**	0.0235	0.6036
+FrameNet	**0.6741**	0.0119	**0.5676**	0.0257	**0.6104**
+FrameNet/Wikidata	0.6630	0.0088	0.5592	0.0164	0.5946

Table 3.12: Mean and max accuracy results over the ten runs on the argument reasoning comprehension development and test sets (best results highlighted in bold).

annotation of the argument reasoning comprehension data leads to more frames per sentence (6.6 on average) than entities per sentence (0.7 on average).

We apply the *late fusion* strategy to aggregate the word, event and entity information: we obtain the annotations separately and combine them afterwards by appending the frame and entity embeddings to the word vectors on the token level. Each input sentence (a claim, a reason or a warrant) is processed by a bi-directional LSTM network that reads not only word embeddings, but also frame embeddings for all event mentions and entity embeddings for all entity mentions. Now, the attention weights for the decision between the two warrants are adapted based on the semantically enriched representation of the claim and the reason.

We optimize hyper-parameters on the development set with random search. All models are trained using the Adam optimizer (Kingma and Ba, 2014) with a batch size of 16. For our evaluation, we perform ten runs and report the mean and max accuracy together with the standard deviation. We report our results for the baseline and three model variants in Table 3.12.

The approaches that use FrameNet and Wikidata information about frames and entities separately improve the averaged performance by more than one percentage point against the baseline model. At the same time, the straightforward combination of the two external knowledge sources, '+FrameNet/Wikidata', does not lead to further improvements. It seems that the current '+FrameNet/Wikidata' model cannot make the full use of the available external information and the results point out the need for advanced models that are able to fuse annotations of different types. Albeit positive, the results do not seem be a strong support for the hypothesis of Habernal et al. (2018a) about external knowledge being beneficial for this task, as we observe only moderate improvements. Overall, the enriched models (+Wikidata, +FrameNet and +FrameNet/Wikidata) make mostly the same predictions as the baseline system. For instance, for +Wikidata there is 79.5 % overlap of the predictions with the baseline, and for +FrameNet, it is 76.6 %. In the following section, we try to identify reasons why the event and entity knowledge does not further improve the results on the argument comprehension task.

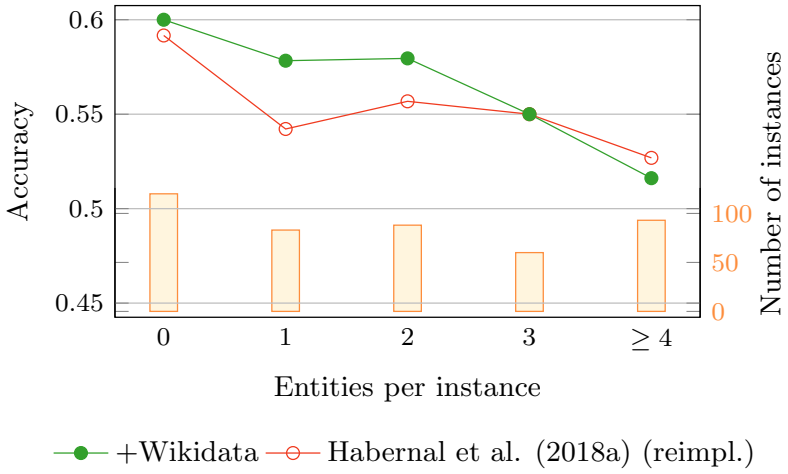

Figure 3.18: Accuracy for the model enriched with Wikidata entity information, '+Wikidata', and for the baseline model, Habernal et al. (2018a) (reimpl.), by the number of entities in one test instance in the argument reasoning comprehension dataset (left axis). The number of test instances in each group is on the right axis.

Error analysis

The amount of semantic information that the model can utilize is dependent on the number of annotations for the input text. We analyze the performance of the enriched models by the number of new annotations for +Wikidata and for +FrameNet.

Figure 3.18 shows the performance of the model enhanced with entity information, +Wikidata, against the number of Wikidata entities per test instance. For comparison, we also plot the performance of the baseline model. As expected, there is no difference in performance for the instances without Wikidata entity annotations (the leftmost column in Figure 3.18). We can see a clear improvement for the instances with one or two recognized entities, which indicates that some semantic knowledge is helping to draw the decision between the two warrants. +Wikidata performs equal to the baseline for three or more annotations.

The distribution of results by the number of events in a test instance for the model enhanced with event information (+FrameNet) shows no clear trend (see Figure 3.19). Once some frame annotations for events are available the enriched model outperforms the baseline, whereas with the growing number of frames the difference in performance decreases as apparently more noise is added to the input. One reason for it might be that both annotation tools, the entity linker and the frame identifier, introduce some noise themselves: for the entity linker, we have already reported around 0.73 F-score on a question answering dataset and the frame identifier has an accuracy of 0.89 on the frame annotated data (Botschen et al., 2018a).

To get more insight into the Wikidata entity annotations for argument reasoning, we perform a manual error analysis on 50 instances of the test set. In 44 % of

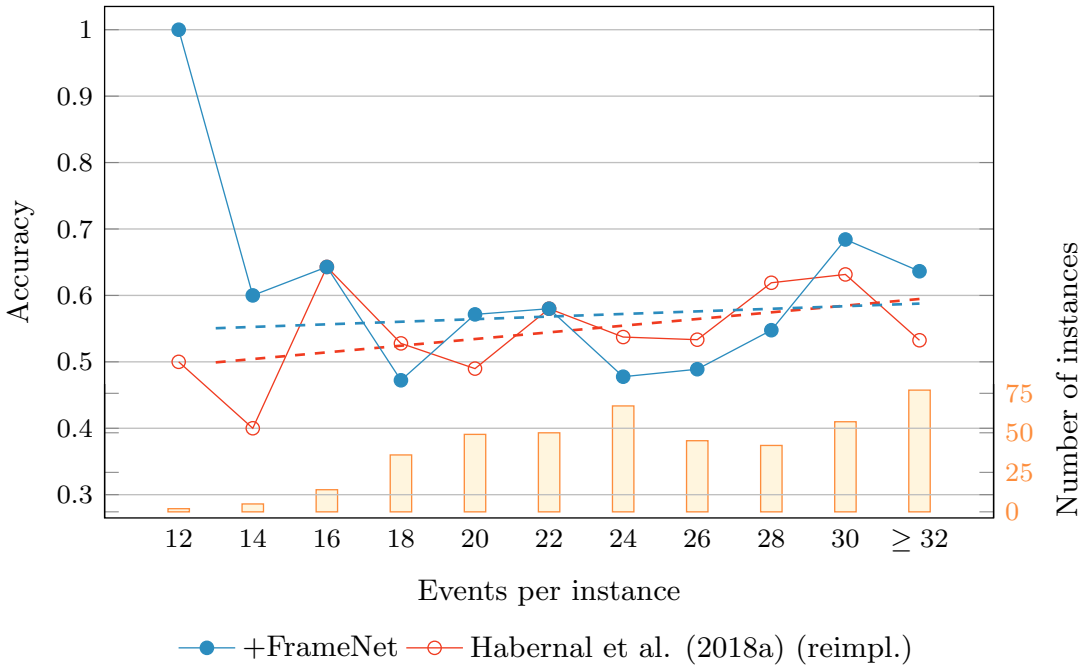

Figure 3.19: Accuracy for the FrameNet approach, +FrameNet, and for the baseline model, Habernal et al. (2018a) (reimpl.), by the number of events (frames) in one test instance in the argument reasoning comprehension dataset (left axis). The number of test instances in each group is on the right axis. 12 is the minimum number of frame annotations in a test instance and there are two such test instances.

the model errors, no Wikidata annotation was available and in 52 %, the entity annotations were (partially) incorrect. Only 4 % of the model errors occur despite correct Wikidata entity annotations being available to the model. It seems that for test instances with one or two Wikidata entity annotations, the semantic enrichment helps to capture a broader context of the argumentation chain, which in turn benefits the classification.

Overall, we can conclude that more annotations with frames or entities lead to more noise and the benefit of the additional external input is reduced. When manually inspecting the annotations of frames and entities, it becomes apparent that the provided background knowledge is not sufficient to draw the distinction between the two warrants. The key difference between the two warrants is often negation, as in:

(i) warrant: they do not have to satisfy customers to make a profit;

(ii) warrant: they have to satisfy customers to make a profit.

Even though our approach performs a semantic enrichment of the context, the crucial capability of performing reasoning is still missing.

Discussion

Our results offer a first perspective on using external resources for the argument reasoning comprehension task. FrameNet and Wikidata are open-domain resources and our enrichment approach is task-independent. We show that it is possible to successfully enrich the context with world knowledge from the two information sources. At the same time, our approach is still limited as to how much it can advance the argument reasoning comprehension task. Despite the analysis of the shared task provided in Habernal et al. (2018a), the key challenge of this task might not be the lexical-semantic or the world knowledge gap between the warrants but rather the modeling of negation and other logical phenomena.

Yet another direction takes the system of Choi and Lee (2018) that won the argument reasoning comprehension shared task (Habernal et al., 2018b). The shared task was conducted concurrently with our research. Choi and Lee (2018) transfer implicit inference knowledge by pre-training on a much larger Stanford Natural Language Inference dataset (Bowman et al., 2015) and fine-tuning the model on the argument reasoning comprehension task. Niven and Kao (2019) apply the modern large pre-trained BERT architecture (Devlin et al., 2019a) to the argumentative reasoning task and find that it can be very effective by exploiting dataset artifacts. Similarly to the previous models, BERT struggles to correctly capture the effect of negation on argumentative chains. Niven and Kao (2019) suggest an adversarial dataset, on which all current models do not outperform a random baseline and which provides a more robust assessment of argument comprehension. The authors further suggest that an adversarial dataset might be a good benchmark for our +FrameNet and +Wikidata architectures (as published in Botschen et al., 2018b), as those could be standing out compared to other models due to the inclusion of some of the required world knowledge. We find adversarial evaluation and negation modeling exciting directions for future work on argumentative reasoning.

3.3.2 Application: Fact Extraction and Verification

Automatic fact verification is inherently tied to using the external knowledge and is a natural application of the knowledge base linking methods. The task is to extract a fact from an input text and to compare it to a repository of world knowledge to verify the truthfulness of the fact. We have already given two examples, how relation extraction (Figure 3.4) and entity linking (Figure 3.12) can be relevant for fact verification.

Fact checking is of utmost relevance in the modern world. The amount of misinformation and factual inaccuracies on the web has increased in the last years and more research efforts have been devoted to automate the process of fact checking (Konstantinovskiy et al., 2018). The recent shared task on Fact Extraction and VERification (FEVER)[16] formulated the necessary steps to verify simple facts

[16] `http://fever.ai/task.html`

that are given in a text form (Thorne et al., 2018). The FEVER shared task uses Wikipedia as the external world knowledge database and formalizes the verification of a factual claim as inference on an natural language input text and Wikipedia article snippets. To verify a fact, one has to determine if Wikipedia contains an information relevant to the given fact and if this knowledge contradicts, supports or does not give enough information about the fact. The sub-tasks defined by Thorne et al. (2018) are:

1. **Document retrieval:** Given a claim, find Wikipedia articles containing information about this claim.

2. **Sentence selection:** From the retrieved articles, extract evidence in the form of sentences that are relevant for the verification of the claim.

3. **Recognizing textual entailment:** On the basis of the collected sentences (evidence), predict one of three labels for the claim: *Supported*, *Refuted*, or *NotEnoughInfo*.

A claim is considered as correctly verified if the right evidence set was retrieved and the correct label was predicted. This task formalization can be used with any textual documents. Thorne et al. (2018) use only Wikipedia articles as documents to construct the FEVER dataset.

The FEVER shared task organizers have constructed an English dataset for fact verification following the above task definition (Thorne et al., 2018). The dataset contains a list of claims and multiple alternative evidence sets for each of them, including the name of the Wikipedia article where the evidence comes from. The authors have manually annotated each claim with an entailment label, if this evidence set supports the given claim or refutes it. The dataset contains 185,445 claims generated based on the 2017 English Wikipedia dump. We show an example of a claim, which is refuted by Wikipedia evidence in Listing 3.2.

We can see from the task definition that fact verification is performed using natural language inference and external knowledge is used in the textual form of Wikipedia articles. Yet, methods using structured Wikidata knowledge are still directly relevant for the information extraction steps of the FEVER shared task pipeline. The document retrieval step requires matching a given claim with the content of a Wikipedia article. Claims in the FEVER shared task dataset frequently feature one or multiple entities that form the main content of the claim. For instance, the example in Listing 3.2 mentions one entity: 'Lorelai Gilmore'. The document retrieval step can be framed as an entity linking problem (Cucerzan, 2007): identify entity mentions in the claim and then map each of the entities to a Wikipedia article about that entity. Afterwards, the linked Wikipedia articles can be used as the set of the retrieved documents for the subsequent steps of the fact verification pipeline.

```
{
  "id": 78526,
  "label": "REFUTES",
  "claim": "Lorelai Gilmore's father is named Robert.",
  "evidence": [
      [
          [<annotation_id>, <evidence_id>, "Lorelai_Gilmore",
              3]
          // "Lorelai has a strained relationship with her
             wealthy parents, Richard and Emily."
      ]
  ]
}
```

Listing 3.2: A snippet of the FEVER fact verification dataset in the JSON format. Here a single claim is shown, the field 'evidence' contains a reference to the Wikipedia article about the entity 'Lorelai Gilmore'. The evidence contains lists of four elements with the dataset ids for annotation and evidence, the Wikipedia article id and the sentence number in the article. The sentence number (3 in this example) is pointing towards the sentence in the article that refutes the claim (see the 'label' field). The evidence text is not part of the dataset, but we include it here as a comment after '//'.

Linking Claims to a Knowledge Base

Similarly to the question answering data discussed in Section 3.2.2, the FEVER dataset contains entities of different categories, such as actor names, locations, movie titles and professions. We apply the developed VCG entity linker for Wikidata from Section 3.2 on the claim. Given the large dataset size, we employ additional heuristics to pre-filter mention candidates and reduce the computational overhead of running the entity linker. Instead of starting with any possible token n-gram, we exclude those n-grams that include the main verb in the sentence (according to the part-of-speech tags)[17]. For example, from a claim "Down With Love is a 2003 comedy film." we would consider 'Down With Love' and 'a 2003 comedy film' as possible entity mentions, but not 'Love is a 2003 comedy'. Although this heuristic would exclude some correct entity mentions containing verbs, it has proven itself effective on the FEVER dataset. Apart from this modification for mention extraction, we do not change or re-train the VCG entity linking model on the new data. We use VCG to link entity mentions to Wikidata entities and to compute the global entity assignment for the claim. To complete the document retrieval step, we convert the Wikidata IDs to Wikipedia article names using the same Wikidata-Wikipedia mapping $Q \mapsto W$ that we have used in Section 3.1.2.

[17] We employ AllenNLP (Gardner et al., 2018) to annotate claims with part-of-speech tags.

Upon the manual inspection of the dataset, we recognized that many Wikipedia titles appear verbatim in the claims. For example, references for events often exactly match the article name (e.g. 'First inauguration of Barack Obama'). That might be an artifact of how the shared task dataset was created. The annotators employed by Thorne et al. (2018) were asked to generate a claim based on a randomly chosen sentence from one of the 50,000 most popular Wikipedia pages. The claim should have contained a single fact and should have been focused on the entity that the original Wikipedia page was about. Thus, the annotators might have been biased to include the article name into the claim.

Given this observation, we devise a soft upper bound method for the entity linking that is based on the lexical overlap between a claim and the popular Wikipedia article names. We employ the constituency parser from AllenNLP (Gardner et al., 2018) and extract all noun phrases from the input claim. We take all noun phrases and the words before and after the main verb as possible article name mentions. We use the MediaWiki API[18] to search through the titles of all Wikipedia articles for matches with the potential verbatim article name mentions found in the claim. The MediaWiki API uses the Wikipedia search engine to find matching articles. The top match is the article with the largest overlap between its title and the query. For each verbatium article mention in the claim, we store up to seven highest-ranked Wikipedia search results. We also perform an exact search over all Wikipedia article titles in the 2017 English Wikipedia dump provided with the shared task dataset and add these results to the set of the retrieved articles. This method is a soft upper bound for entity linking on the FEVER fact verification task, as it assumes that the name of the article describing the entity will appear verbatim in the claim.

Entity Linking for Factual Claims Evaluation

We compare our entity linking approach (VCG), the soft upper bound solution based on the MediaWiki API exact matching and the baseline document retrieval model provided for FEVER by Thorne et al. (2018). We follow Thorne et al. (2018) and compute accuracy to evaluate the document retrieval step. If the label of the claim is *Supported* or *Refuted* then the retrieved document set has to fully contain the correct set of evidence documents. If the correct label of the claim is *NotEnoughInfo* then it is always considered correct. The document retrieval accuracy value we compute here reflects the best possible result one could achieve for the later textual entailment step with the given document retrieval method (see Section 5.3 in Thorne et al., 2018).

Table 3.13 shows the results by the number of retrieved Wikipedia article (N) per claim in the FEVER dataset. For each method, we rank all retrieved documents for a claim by model confidence and take the top N. For the entity linking methods, it is a ranked list of entities identified in the claim, where each entity is mapped to a Wikipedia article. Some claims in the FEVER dataset need evidence from

[18] https://www.mediawiki.org/wiki/API:Main_page

	Top N			
	1	2	3	5
Thorne et al. (2018)	0.502			0.702
VCG entity linker	**0.660**	0.720	**0.737**	
MediaWiki exact match (soft upper bound)	0.678	0.877	0.926	0.933
Hard upper bound	0.879	1.000	1.000	1.000

Table 3.13: Document retrieval accuracy on the FEVER shared task dataset by the number of retrieved documents (N). Some claims require evidence from multiple documents and thus the hard upper bound at $N = 1$ is lower than 1.0.

multiple Wikipedia article and thus the absolute possible upper bound for the document retrieval accuracy on the FEVER for $N = 1$ is less then 1.0.

Thorne et al. (2018) only report the result for the top retrieved document ($N = 1$) and for the top five ($N = 5$). Our VCG entity linker does not extract more than three entities for any claim in the dataset, so we report the results for VCG up to $N = 3$ (there is no strict limit on the number of extracted entities per claim, but no claim contains more than three entity mentions according to our entity linker). The VCG entity linker proves effective on the factual claims and outperforms the baseline from Thorne et al. (2018) by a wide margin both for $N = 1$ and $N > 1$. The document retrieval accuracy for the VCG entity linker for $N = 2$ is already higher than for Thorne et al. (2018) for $N = 5$. The results show that VCG improves the document retrieval accuracy for different number of retrieved documents N. We note that the VCG entity linker was applied off-the-shelf and was not re-trained or fine-tuned on the fact checking data.

Comparing to the soft upper bound of the MediaWiki API exact match, we see that VCG almost matches it for $N = 1$. The MediaWiki API exact match method relies on the fact that many Wikipedia article titles appear verbatium in the FEVER dataset and therefore, the MediaWiki API method would not be applicable on other data and in a different fact verification scenario. We use it here as a realistic upper bound for our entity linking model. We can see that for $N = 3$ there is still a lot of room for improvement between the VCG model and the MediaWiki API method. Overall, this extrinsic evaluation shows that our entity linking approach can be directly applied to fact verification data that include world entities.

The manual analysis of the entity linking errors for VCG on the FEVER dataset shows that they can be broadly divided into three classes: (1) Spelling errors: a word in the claim or in the article title is misspelled. E.g. "Homer Hickman wrote some historical fiction novels." vs. 'Homer Hickam'. These are dataset errors. (2) Missing entity mentions: a claim does not mention an entity that would correspond to a relevant Wikipedia document. In this case, our assumption that a claim contains at least one entity is wrong. (3) Disambiguation failures: these are the errors of the entity linker. For example, "Anderson Cooper is apolitical." refers

to the entity 'Anderson Cooper (the CNN host)', which requires the knowledge of media landscape to be identified correctly.

FEVER Shared Task Evaluation

Hanselowski et al. (2018) give a detailed description of our system that was submitted to the FEVER shared task system. We provide a short overview of the architecture and present evaluation results on the FEVER development set here.

We use our ad-hoc MediaWiki API exact match solution for document retrieval in our team's submission to the FEVER competition. It delivers the best result on this dataset (see the row for 'MediaWiki exact match (soft upper bound)' in Table 3.13), relying on the particular property of the data. It is important to remember that this MediaWiki API document retrieval method is fitted to the FEVER dataset and will not transfer to fact verification data that was created with a different process.

The sentence selection and the textual entailment components are based on the Enhanced Sequential Inference Model (ESIM) architecture (Chen et al., 2017). ESIM was devised for the natural language inference task of determining entailment between two statements. It creates a rich representation of statement-pairs in three consecutive steps: input encoding, local inference modeling and inference composition. A bidirectional LSTM (BiLSTM) is used to compute input token encodings. Then, an attention matrix for each sentence is computed over the other sentence in the pair and another token encoding is computed from the attention weights and the previous token encodings. Finally, both the original token encodings and the attention token encodings are fed into two BiLSTMss to compute updated token representations. Maximum and average pooling are applied to derive two sentence vectors, which are then concatenated and fed into a multilayer fully-connected network for the classification decision. We refer to the description of ESIM in Chen et al. (2017) for further details.

For sentence selection, we train the ESIM architecture to predict a relevance score for two sentences (the claim and a sentence from the retrieved documents). As a result, we are able to rank all sentences of the retrieved documents according to the computed relevance scores. We select the five highest-ranked sentences as the evidence set for the recognizing textual entailment step.

To classify the input claim into the one of the three entailment classes (*Supported*, *Refuted* or *NotEnoughInfo*), we compute the entailment between the retrieved evidence set and the claim. We again use the ESIM architecture as a basis for our solution. The ESIM architecture for textual entailment in Chen et al. (2017), as well as our ESIM-based sentence selection architecture above take two sentences as an input. To process the evidence set and the claim together, we devise an ESIM-based model that is able to process up to five evidence sentences and a claim. To enable that, we combine the claim with each sentence from the evidence set and feed the resulting pairs separately into an ESIM model and process

Task	Metric	System	Score
1. Document retrieval	accuracy	baseline	0.702
		our system	0.936
2. Sentence selection	recall	baseline	0.442
		our system	0.871
3. Recognizing textual entailment	label accuracy	baseline	0.521
		our system	0.685
Full pipeline	FEVER score	baseline	0.323
		our system	**0.647**

Table 3.14: Performance comparison of our fact verification system and the baseline system on the FEVER shared task development set.

them in parallel. The last hidden states of the five individual claim-sentence runs of ESIM are compressed into one vector using attention and pooling operations. The final representation is fed into a three-way classification layer. The intuition behind our architecture is to allow the model to extract information from the five sentences that is most relevant for the final fact verification decision. We use the GloVe (Pennington et al., 2014) and FastText (Bojanowski et al., 2017) embeddings as word representations for both claim and evidence sentences.

In Table 3.14, we report the performance of the three sub-systems of our fact verification pipeline for the FEVER shared task and the final overall score. We compare to the baseline system of Thorne et al. (2018) on the development set that was released by the shared task organizers. The FEVER score for the full pipeline considers the entailment prediction correct only if the correct supporting documents and the evidence set were retrieved as well (see Thorne et al., 2018). The evaluation results demonstrate that we were able to improve upon the baseline on each sub-task. The performance gains over the whole pipeline add up to an improvement of 100 % with respect to the baseline system.

Discussion

In this section, we have discussed the application of the entity linking to fact verification and presented a system for the participation in the FEVER shared task (Thorne et al., 2018). We have shown that our VCG entity linker can be applied off-the-shelf on fact verification data. Our full system for fact verification (first described in Hanselowski et al., 2018) has ranked third in the final evaluation of 23 competing systems[19].

Recently, our fact verification system was further evaluated by the organizer of the follow-up FEVER 2.0 shared task for its resistance to adversarial attacks. Thorne

[19] See 'Athene UKP TU Darmstadt' in FEVER shared task leaderboard: `http://fever.ai/2018/task.html`

et al. (2019) analyzed the participating systems from the first edition of the shared task and applied them on adversarial examples. The overall performance of our system has degraded dramatically under this type of attacks, which was a common trend among all the tested systems. However, the document retrieval component has been more robust. The recall of the evidence retrieval component of our system has degraded by -0.05 compared to the averaged degradation of -0.11 among the other tested systems (see Table 5 in Thorne et al., 2019).

3.4 Chapter Summary

Relation extraction and entity linking can be used to construct a full grounded semantic graph representation of an utterance (as we visualize in Figure 2.11). This chapter addressed the first research question that we have formulated for the present dissertation:

RQ1 What are the challenges for linking a text to an external knowledge base and how can the linking methods be improved?

We have shown that existing methods for relation extraction and entity linking lack a comprehensive modeling of the surrounding context. We have developed our linking methods to incorporate the contextual information on different modelling levels. At the same time, downstream applications were the additional driving motivation for the task definitions and the methods, which we proposed in this chapter. We have used question answering and fact verification tasks to guide the development process for relation extraction and entity linking.

In the first part of this chapter, we have introduced a neural network architecture for relation extraction on the sentence level that takes into account other relations from the same context. We have shown by comparison with competitive baselines that these context relations are beneficial for relation extraction with a large set of relation types. Our approach can be easily applied to other types of relation extraction models as well. For instance, Lin et al. (2016) extract sentence-level features and then combine features from multiple sentences with a selective attention mechanism. It would be possible to replace their sentence-level feature extractor with our model. Bekoulis et al. (2018) show that adversarial training can be used to make relation extraction models more robust across different domains and languages. In the future, we could extend our distant supervision dataset construction method to other languages than English and to apply it on other domains than Wikipedia. That would allow us to broaden the evaluation of the proposed context-aware relation extraction model. The adversarial method of Bekoulis et al. (2018) adds noise to existing training data instances and would be directly compatible with our training setup.

In the second part of the chapter, we have described the task of entity linking on short and noisy question answering data and its challenges. The suggested new approach for this task is a unifying network that models contexts of vari-

able granularity levels to extract features for joint mention detection and entity disambiguation. This system achieves state-of-the-art results on two datasets and outperforms the previous best system used for entity liking in the question answering task. The results further verify that modeling different context granularity levels achieves a better performance across various entity categories (cf. the macro F-score results in Table 3.8 and per category results in Figure 3.16).

In the parallel line of work, Yu et al. (2017) have attempted to incorporate entity linking and relation extraction into a single model by aggregating the lists of predicted relation and entity candidates and by re-ranking them together. It would be possible to aggregate the predictions of our relation extraction and entity linking into a list of entity and relation combinations in the same input and apply a global entity-relation re-ranking model. Peng et al. (2017) unify the set of the possible entities and the set of the possible relations into a single label set for the model to predict and therefore learn only one large model for both entity linking and relation extraction. This offers an exciting future direction for our entity linking model. The learnings from the entity linking experiments in this chapter can be incorporated into the semantic parsing model that we present in Chapter 4 by making it select from the joint entity and relation label set.

Our linking methods advance the state of the art on the respective benchmarks and enable more efficient downstream applications. Thereby, we answer the RQ1 with the new context-aware method for the two core grounding tasks: relation extraction and entity linking. The presented methodological contributions form the basis for our work on semantic parsing and knowledge base question answering that we describe in the next chapter. Furthermore, our entity linking models have been applied to enhance NLP pipelines with world knowledge, as we have sketched out in Section 3.3. The successful extrinsic evaluation of the entity linker on argument comprehension and fact verification tasks forms the basis for our answer to the RQ3. Our code, trained models and the used dataset splits for both relation extraction and entity linking are freely available online.[20]

[20] Both repositories can be found under `https://github.com/UKPLab/`.
Relation extraction: `https://github.com/UKPLab/emnlp2017-relation-extraction`
Entity linking: `https://github.com/UKPLab/starsem2018-entity-linking`

Chapter 4

Knowledge Base Question Answering

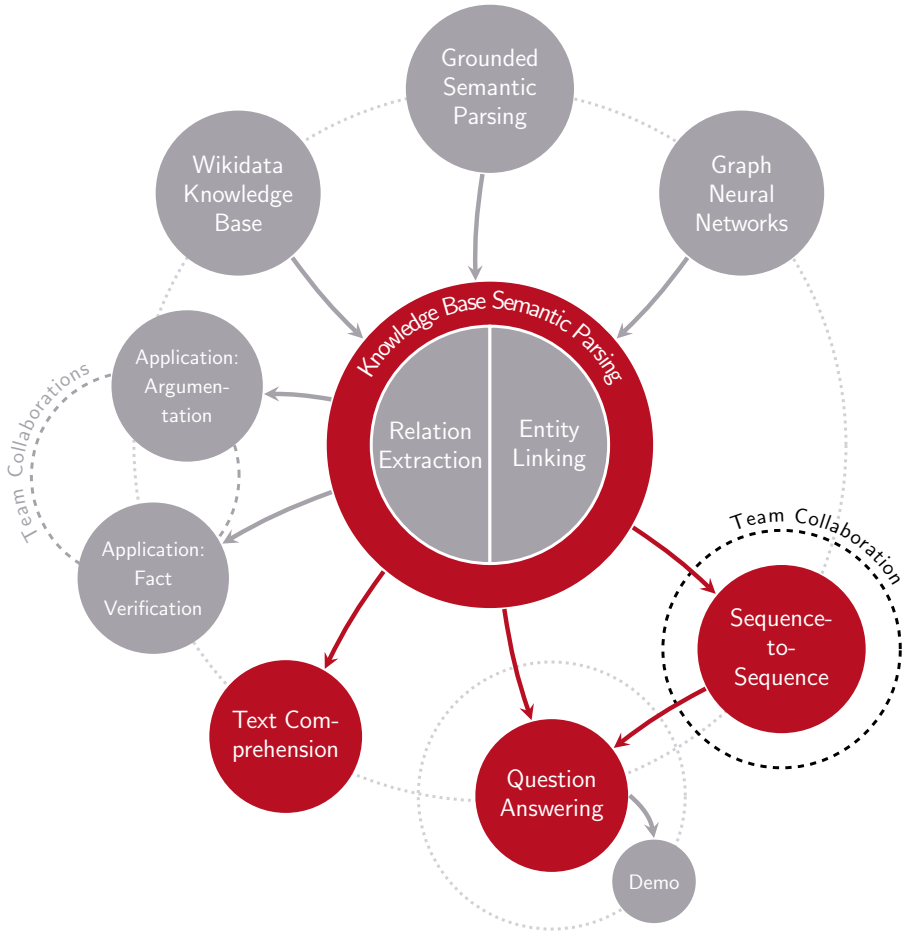

Figure 4.1: Chapter 4 in the thesis diagram. This chapter discusses the combination of the linking methods into a semantic parser and its application to question answering and text comprehension (the large red circle at the center and the red circles in the bottom). We contrast our semantic parser with a sequence-to-sequence model in a team collaboration project.

In this chapter, we turn to the question answering task and to how it can be approached with grounded semantic parsing that uses Wikidata. Semantic parsing has been a commonly used formalism and the main building block for successful knowledge base question answering systems (Yih et al., 2015 and Reddy et al., 2016 among others).

We present our architecture for mapping a question to the structure of the knowledge base and for constructing a grounded semantic representation. We focus on the problem of learning numerical vector representations for complex semantic graphs that consist of multiple entities and relations. Previous work on grounded semantic parsing has been concerned with selecting the correct semantic relations for a question and disregarded the structure of a semantic representation: the connections between the entities and the directions of the relations. In our approach, we process the structure, the entities and the relations together in one model. As our main contribution, we propose to use Gated Graph Neural Networks (GGNNs) and adapt them to encode the graph structure of the semantic representation. We show on an open-domain question answering dataset that GGNN-based methods outperform all baseline models that do not explicitly model the semantic structure. Thereby, this chapter addresses the second research question that we have formulated in the introduction:

RQ2 How to encode structured semantic representations for effective grounded semantic parsing?

After demonstrating the effectiveness of GGNNs, we combine our GGNN-based semantic parsing method for knowledge base question answering with a text comprehension system. We apply our semantic parser on a large text comprehension dataset, where the goal is to retrieve an answer from textual sources. In this setting, our system verifies the answers retrieved by a text comprehension model with knowledge base information. The knowledge base coverage is limited on text comprehension data. We examine how the grounded semantic parsing system behaves when it is much harder to model the input using the knowledge base. In that regard, this chapter contributes further results for the third research question, which we have already considered in Chapter 3:

RQ3 Can the knowledge base linking methods and the grounded semantic parser generalize to new data and be embedded into natural language understanding applications?

Finally, we discuss an orthogonal approach to the representation learning with GGNNs. We report on a collaboration project that has attempted to construct a fully end-to-end knowledge base question answering architecture with sequence-to-sequence networks. Such models require much more training data and we show that it is possible to bootstrap a sequence-to-sequence model with synthetic input. Albeit positive, the performance of our sequence-to-sequence model does not surpass the GGNN semantic parsing architecture.

This chapter is organized into six sections. Sections 4.1 through 4.3 introduce the knowledge base question answering task, our motivation, the grounded semantic parsing approach and the evaluation methodologies that we use for the proposed methods. Section 4.4 introduces the representation learning framework to process semantic representations for questions using GGNNs. In this section, we describe our main contribution: the GGNN model architecture and the results on the

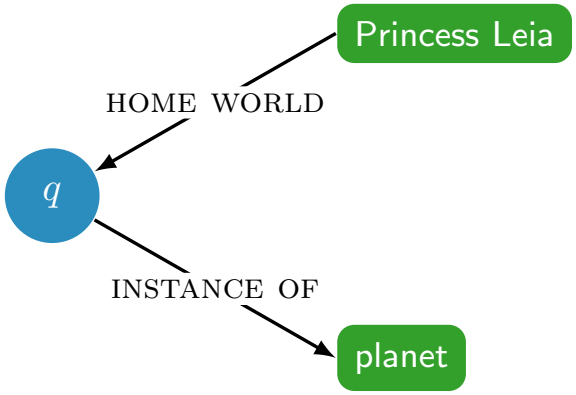

Figure 4.2: A semantic graph for the question "What is Princess Leia's home planet?". The graph features two entities and a variable q which are interconnected with two relations. q stands for the entity that is referenced in the question, but which identity is unknown.

knowledge base question answering data. In Section 4.5, we apply our grounded semantic parsing approach on the text comprehension task and combine it with a model that selects an answer span from a text document. In the last section of this chapter (Section 4.6), we consider an alternative to the representation learning framework and build a sequence-to-sequence solution for mapping questions to the knowledge base.

4.1 Task Description and Motivation

We start by introducing the knowledge base question answering task. Given a natural language question, the task is to find an entity or a set of entities in the knowledge base that constitutes the answer. For example, for the question "What is Princess Leia's home planet?" the answer, "Alderaan", could be retrieved from a general-purpose knowledge base, such as Wikidata. We have discussed already in Section 2.1 how a knowledge base can function as a universal storage model for the information about the world. Hence, a successful knowledge base question answering system would ultimately provide an easily accessible natural language interface to factual knowledge (Liang, 2016).

Question answering requires precise modeling of question semantics through the entities and relations available in the knowledge base to retrieve the correct answer. Figure 4.2 shows how the above example could be modeled using Wikidata. The depicted graph structure consists of entities and relations from the knowledge base and a special q-node that denotes a free variable. It stands for the entity or entities that are being referenced in the question. With the interpretability of the semantic representation defined in Section 2.2.2, we are able to retrieve the set of entities for q from Wikidata, for which the relations in the graph hold. The answer to the question will contain any entity from the knowledge base that can take the place of the q-node and satisfy all the relations in the graph.

Modern semantic parsing approaches usually focus either on the syntactic analysis of the input question (Reddy et al., 2016) or on detecting individual knowledge base relations (Yu et al., 2017), whereas the structure of the semantic representation is ignored or only approximately modeled. Reddy et al. (2016) use the syntactic structure of the question to build all possible semantic representations and then apply a linear model with manually defined features to choose the correct representation. Only a subset of their manually designed features encodes some basic information about the structure of the semantic representation, such as the number of nodes in the candidate semantic graph. These features do not encode the complete graph and dependencies between the nodes.

The state-of-the-art approach for knowledge base question answering of Yih et al. (2015) and Bao et al. (2016) restricts all output semantic representations to a single semantic relation, which they call a core relation, and a small set of constraints that can be added to it. Their system uses manual features for the constraints and a similarity score between the core relation and the question to model the semantic representation. By focusing on the core relation, their system limits the search space of possible question interpretations, but ultimately it cannot handle complex questions that require a composition of two or more knowledge base relations into a semantic representation to find the answer. Luo et al. (2018) further extend the semantic parsing approach of Yih et al. (2015) and treat each constraint and edge in the graph in the same way as the core chain. They apply a neural encoder on each component of the semantic representation and use an average of the resulting vectors to aggregate them into a single encoding. Their approach thus puts equal focus on different graph components, but fails to encode the structure of the graph of the semantic representation. They show a moderate improvement in their evaluation, but fail to outperform Yih et al. (2015).

The aforementioned systems were evaluated on the WebQuestions dataset (Berant et al., 2013). It is instructive to look at the complexity of the semantic graphs in the dataset: Figure 4.3 plots results for the state-of-the-art systems by the number of relations that needs to be identified to get the correct answer to a question. We include results for the semantic parsing systems that have published the output on the WebQuestions dataset. We compute the number of required relations from the manually annotated Freebase semantic representations provided by Yih et al. (2016) for the same dataset. For example, the question in Figure 4.2 requires two relations to find the answer. It can be seen in Figure 4.3 that the lack of structure modeling in the modern approaches results in a worse performance on more complex questions that require more than one relation.

Our analysis shows that the current solutions for knowledge base question answering do not perform well on complex questions. These observations motivate us to propose an approach that puts the structure of a semantic representation at the center of the model. We claim that one needs to explicitly model the semantic structure together with the disambiguation and linking step to find the correct semantic representation for complex questions. We address this gap with

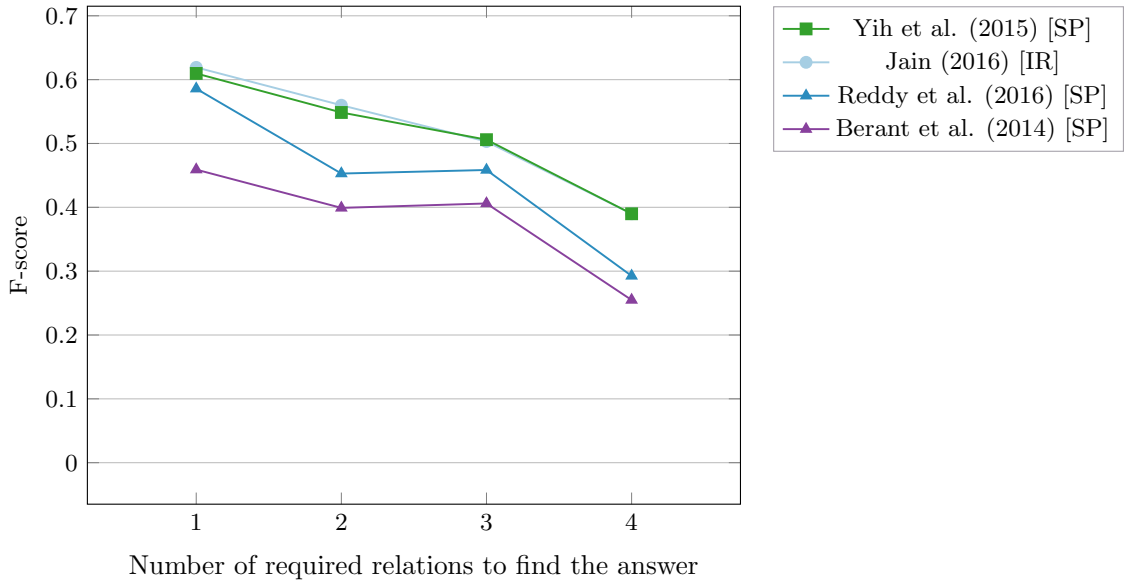

Figure 4.3: Question answering results on the popular WebQuestions dataset from the previous work by the number of relations needed to find the correct answer. Those systems used the Freebase knowledge base and are not directly comparable to our Wikidata methods. The diagram includes published results for question answering systems at the time of the experiments. [SP] marks semantic parsing based systems, and [IR] is for information retrieval approaches.

our work and investigate ways to encode the structure of a semantic representation and to improve the performance for more complex questions. We adapt GGNNs which we have introduced in Section 2.3 to process and score semantic representations. We apply GGNNs on directed graphs with labeled edges and adapt them to handle a large set of possible entities and relation types from the knowledge base, as the original GGNN architecture is only suitable for unlabeled relations and nodes. To verify empirically that GGNNs offer an improvement over non-graph architectures, we construct a set of baselines from previous work. We train and evaluate the baseline models and our proposed architecture in the same controlled question answering environment. Throughout the experiments, we use Wikidata to construct grounded semantic representations and retrieve the answers.

In parallel to this research, Sun et al. (2018) have conducted a study on using GNNs in the context of document question answering. The authors explore the combination of external knowledge and text to answer open-domain questions. They link knowledge base statements to document paragraphs and use GNNs to score each entity in the linked statements as an answer or not. They do not construct an explicit semantic representation or a graph to encode the question and hence do not operate in the semantic parsing framework. To the best of our knowledge, we are the first to use GNNs for semantic parsing and knowledge base question answering. In Section 4.5, we also explore the semantic parsing application for question answering on documents.

4.2 Semantic Parsing for Question Answering

We describe a semantic parsing approach to knowledge base question answering. For each input question, we construct an explicit structural semantic representation as in Figure 4.2. Since we represent the semantics of the input by mapping it to the knowledge base, the resulting semantic representation is always a *semantic graph*, which is a subgraph of the knowledge base[1]. Such semantic graphs can be deterministically converted to a SPARQL query to extract the answers from the knowledge base (Figure 2.11) and hence a semantic graph is also a graphical representation of the knowledge base query. Similar graphical representations for semantic parsing were used in the previous work (Yih et al., 2015; Reddy et al., 2016; Bao et al., 2016; Luo et al., 2018).

An alternative solution to semantic parsing is to build an information extraction pipeline that views the question as a query and the knowledge base as a source of relevant information (Yao et al., 2014). Dong et al. (2015) and Jain (2016) construct a vector representation for the question and use it to directly score candidate answers from the knowledge base without semantic parsing. However, such approaches are hard to analyze for errors or to modify with explicit constraints as opposed to our approach that constructs a semantic representation. For example, it is not directly possible to implement the temporal sorting constraint (cf. argmax below). Dong et al. (2015) and Jain (2016) have observed that their information extraction systems for question answering suffer from errors related to temporal or quantitative sorting. The semantic parsing methods have generally achieved better performance on the knowledge base question answering task (Cheng et al., 2017). Therefore, we regard only semantic parsing methods in our work.

Knowledge base question answering is a well-established semantic task and the range of machine learning methods that have been applied to it includes those that rely heavily on manually defined features (Berant et al., 2013), neural approaches (Jain, 2016) and syntax-based systems (Reddy et al., 2017). We build upon the systems of Yih et al. (2015) and Bao et al. (2016) that construct explicit semantic representations but do not rely on syntax to do so. Previously, Berant et al. (2013) and Reddy et al. (2017) have constructed semantic representations from the syntactic parse. This aligns the semantic structure with the syntax as it is common in formal semantics (see Section 2.2). Yet, syntactic pre-processing leads to error propagation and over-restricts the space of possible semantic graphs. A knowledge base that has been developed independently of the language does not make the same assumptions about entities and relations as formal semantics and thus cannot be assumed to match the natural language syntax perfectly. We have already discussed this mismatch in more detail in Section 2.2. To summarize, we leave out the 'Syntax' and 'Ungrounded logic' steps in the semantic parsing pipeline in Figure 2.11 and aim to construct the grounded logic-based representation directly from the input sentences and a set of recognized entities. Thereby, we

[1] Not every type of a semantic representation is necessarily a graph, cf. Frame-based representations that we have discussed in Section 2.2.

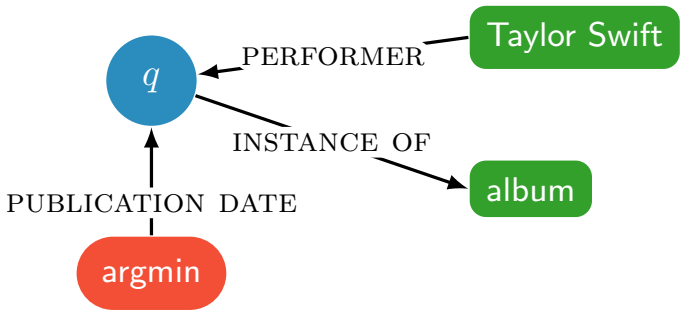

Figure 4.4: A semantic graph for the example question "What was the first Taylor Swift album?". The graph features two entities, a variable q and a constraint node argmin which are inter-connected with three relations. q stands for the entity that is referenced in the question. argmin encodes a restriction on the value of PUBLICATION DATE.

unify the relation extraction and semantic representation construction in a single semantic parsing model.

Semantic graphs We employ structural semantic representations in the form of graphs to encode the meaning of a question. We use the first-order logic based semantic language grounded in Wikidata that we have defined in Section 2.2.2. We describe a subset of the language here that we use for the knowledge base question answering task. Our semantic graphs consist of a question variable node (q), Wikidata entities (e.g. Taylor Swift), relation types from Wikidata (e.g. PERFORMER) and constraints on the date and number values (see Figure 4.4 for an example graph with a constraint). The q-node is always present and denotes the answer to the question. All entities from the knowledge base that can take its place so that all relations and constraints hold, constitute the answer to the question.

We write down a graph as a set of edges $\mathcal{E}$. Each edge links the q-node to another Wikidata entity, a number or a date. The edge set $\mathcal{E}$ is defined via Wikidata relations and entities. Recall that Wikidata can be formally described as a large graph $W = (E, R, I)$, where E is a set of entities, R is a set of binary relation types and I is a collection of relation instances encoded as $\langle e_1, r, e_2 \rangle$ with $r \in R$, $e_1, e_2 \in E$ (e.g. ⟨Hawaii, CAPITAL, Honolulu⟩). Then, the edge set $\mathcal{E}$ of a semantic graph consists of binary edges that correspond to Wikidata relation instances with q in place of one of the relation arguments. For temporal relations, the second argument can be either an argmax or an argmin constraint. This means that only entities that have the maximum or the minimum value of that relation are considered for the answer. This definition represents a subset of the grounded semantic language in Section 2.2.2, and each semantic graph can be rewritten as a first-order logic formula. As mentioned above, the graphs are also always isomorphic to SPARQL queries that can be used to evaluate a graph against the knowledge base.

Algorithm 1 Graph state generation procedure

An empty edge set $\mathcal{E}_0 = \emptyset$
An empty state $s_0 = (\mathcal{E}_0, \mathcal{F}_0)$
Available actions $\mathcal{A} = \{a_e, a_c, a_m\}$
The step counter $t = 0$
An empty set of states $\mathcal{S}_0 = \emptyset$
The set of generated states at the next step $\mathcal{S}_0 \leftarrow s_0$
while $|\mathcal{S}_{t+1}| > |\mathcal{S}_t|$ **do**
    $t = t + 1$
    $\mathcal{S}_{t+1} = \mathcal{S}_t$
    for all $a \in \mathcal{A}$ **do**
        for all $s \in \mathcal{S}_t$ **do**
            $\mathcal{S}_{t+1} \leftarrow a(s)$
        end for
    end for
end while
output: The set of graph states after the last step S_T

Our semantic graphs are inspired by the query graphs of Yih et al. (2015) and their extension in Bao et al. (2016), the key difference being that we allow graphs that have multiple independent relations with additional constraints instead of just one core relation with constraints. We also allow relations to attach to other nodes rather than the q-node enabling longer relation paths. Thus, the semantic graphs defined here are a superset of the query graphs in Yih et al. (2015) and allow us to model more complex questions. Consequently, they also correspond to the formalized representations used by Reddy et al. (2014) and the λ-Dependency-Based Compositional Semantics in Berant et al. (2013), since those were the foundation for the query graphs in Yih et al. (2015).

Semantic graph construction Yih et al. (2015) defined a step-by-step graph generation that does not need full syntactic parses and was also adopted in subsequent publications (Bao et al., 2016; Peng et al., 2017; Luo et al., 2018). We use the same idea as in Yih et al. (2015) to construct semantic graphs step-by-step and without syntax. We add modifications to the process that allow us to build more expressive and complex graphs.

We define a set of states and a set of actions that can be applied at each state to extend the current semantic graph. A state is a tuple of a graph and a set of free entities: $s = (\mathcal{E}, \mathcal{F})$, where the graph is $\mathcal{E}$, as defined above, and $\mathcal{F} = \{e|e \in E\}$. The set of free entities $\mathcal{F}$ contains the entities that were identified in the question but were not yet added to the graph. We use an off-the-shelf entity linker to identify and disambiguate entity mentions in the question: if available for a dataset, we use the same entity linking setup as the previous work, otherwise we apply our entity linker from Section 3.2. The graphs are generated step-by-step

Algorithm 2 The *add edge* action a_e

The q-node
A graph state $s_t = (\mathcal{E}_t, \mathcal{F}_t)$
function $a_e(\mathcal{E}_t, \mathcal{F}_t)$
  $\mathcal{S}_{t+1} = []$
  if $|\mathcal{F}_t| > 0$ **then**
    $e = \text{POP}(\mathcal{F}_t)$
    $R_e = \text{WIKIDATA\_RELATIONS}(e)$
    for all $r \in R_e$ **do**
      $s' = (\mathcal{E}_t \cup \langle q, r, e \rangle, \mathcal{F}_t - \{e\})$
      $s'' = (\mathcal{E}_t \cup \langle q, r, d \rangle, \langle d, r, e \rangle, \mathcal{F}_t - \{e\})$
      $\mathcal{S}_{t+1} \leftarrow s', s''$
    end for
  end if
  return $\mathcal{S}_{t+1}$
end function

starting from an empty graph with no edges that consists only of a q-node. The first state is a tuple of an empty graph and a set of all entities identified in the question $s_0 = (\emptyset, \mathcal{F})$.

Three types of actions can be applied at each step to extend the current graph. Let $\mathcal{A} = \{a_e, a_c, a_m\}$ be that set of actions that can be taken to extend the graph at the current state. The generation procedure is summarized in Algorithm 1. At each generation step, there exists a set of graph states $\mathcal{S}_t$ that was produced after the previous step t. Next, each action from $\mathcal{A}$ is applied to each state s in the current set of states $\mathcal{S}_t$. The newly generated graph states are added to $\mathcal{S}_t$ to produce $\mathcal{S}_{t+1}$. The step-by-step semantic graph generation ends after no new states are produced by the application of actions (that is $|\mathcal{S}_{t+1}| = |\mathcal{S}_t|$).

The action a_e (*add edge*) takes a free entity e from the set $\mathcal{F}$ at the current state s_t. Then, it queries the knowledge base (Wikidata) and retrieves the set of available relation types R_e for the entity e. Since the correct relation is yet unknown, for each relation type $r \in R_e$, it creates a new copy of the graph and adds a new directed edge between the q-node and e with the relation type r: $a_e(\mathcal{E}, \mathcal{F}) = (\mathcal{E} \cup \langle q, r, e \rangle, \mathcal{F}_t - \{e\})$. Contrary to Yih et al. (2015) and Bao et al. (2016), we allow two-step paths between the q-node and the entity e to be added as well: $\langle q, r, d \rangle, \langle d, r, e \rangle$. The action a_e produces new graph states until the set of free entities $\mathcal{F}$ in the input state is empty. Algorithm 2 displays the pseudocode for the a_e (*add edge*) action.

We include the pseudocode for the actions a_c and a_m in Algorithm 3 and Algorithm 4 respectively. The action a_c (*add numeric edge*) follows the same procedure as a_e, but instead of taking a free entity, it searches if the question contains a number or a date and adds a new edge: $a_c(\mathcal{E}, \mathcal{F}) = (\mathcal{E} \cup \langle q, r, n \rangle, \mathcal{F}), r \in R_{d/n}$, where n is a number or a date in the question and $R_{d/n}$ is a set of Wikidata relation types

Algorithm 3 The *add numeric edge* action a_c

The q-node
A number n extracted from the question tokens x
A graph state $s_t = (\mathcal{E}_t, \mathcal{F}_t)$
Wikidata relations that allow dates and numbers $R_{d/n}$
function $a_c(\mathcal{E}_t, \mathcal{F}_t)$
  $\mathcal{S}_{t+1} = []$
  if $\langle q, \cdot, n \rangle \notin \mathcal{E}_t$ **then** ▷ where $\cdot$ is a placeholder for any relation
    for all $r \in R_{n/d}$ **do**
      $s' = (\mathcal{E}_t \cup \langle q, r, n \rangle, \mathcal{F}_t)$
      $\mathcal{S}_{t+1} \leftarrow s'$
    end for
  end if
  return $\mathcal{S}_{t+1}$
end function

Algorithm 4 The *add argmax/argmin* action a_m

The q-node
A graph state $s_t = (\mathcal{E}_t, \mathcal{F}_t)$
Wikidata relations that allow dates and numbers $R_{d/n}$
function $a_m(\mathcal{E}_t, \mathcal{F}_t)$
  $\mathcal{S}_{t+1} = []$
  if $\langle q, \cdot, \text{argmax} \rangle \notin \mathcal{E}_t \wedge \langle q, \cdot, \text{argmin} \rangle \notin \mathcal{E}_t$ **then**
    for all $r \in R_{d/n}$ **do**
      $s' = (\mathcal{E}_t \cup \langle q, r, \text{argmax} \rangle, \mathcal{F}_t)$
      $s'' = (\mathcal{E}_t \cup \langle q, r, \text{argmin} \rangle, \mathcal{F}_t)$
      $\mathcal{S}_{t+1} \leftarrow s', s''$
    end for
  end if
  return $\mathcal{S}_{t+1}$
end function

that allow dates and numbers as values.[2] We allow the a_c action to be applied only once to a graph. Finally, the action a_m (*add argmax/argmin*) adds a new edge with either argmax or argmin sorting constraint on numbers and dates to the semantic graph: $a_m(\mathcal{E}, \mathcal{F}) = \{(\mathcal{E} \cup \langle q, r, \text{argmax} \rangle, \mathcal{F}), (\mathcal{E} \cup \langle q, r, \text{argmin} \rangle, \mathcal{F})\}, r \in R_{d/n}$, where $R_{d/n}$ is a set of Wikidata relation types that allow dates and numbers as values. We allow the a_m action to be applied only once to a graph.

Our semantic graph construction process allows to effectively search the space of possible graphs for a given question through an iterative application of the defined actions $\mathcal{A}$ on the states $\mathcal{S}_t$ generated at the previous step (Figure 4.5). At the same

[2] We do not distinguish between number and date values, as those are often ambiguous. E.g. "who was the prime minister in 2012?"

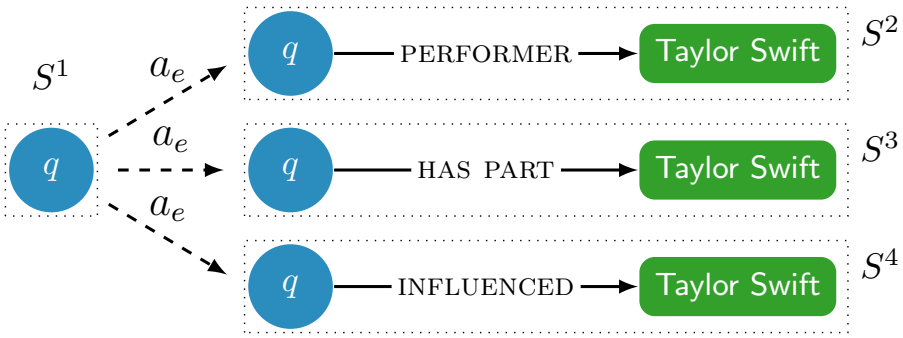

Figure 4.5: A single semantic graph generation step: applying the *add edge* action a_e on an empty graph. The result of the action is a set of three new graphs, one for each possible relation. A statistical model is used to select one correct graph from the set once the generation is completed.

time, the set of allowed actions limits the search space in the presence of large knowledge base graphs. For instance, it means that in practice we always stop the search after all entities recognized in the question were used and the edges are of the maximum length of two. The final set of states contains all graphs that were generated during the application of the procedure. These graphs constitute the set of candidate semantic representations for the given question.

4.3 Evaluation Data and Metrics

We use Wikidata as the primary knowledge base and use it to construct two evaluation datasets. Our primarily goal is to set up experiments that allow us to test if graph networks are better at learning representations of semantic graphs than previous approaches. To enable an evaluation with Wikidata, we prepare the WebQSP-WD dataset to compare the GGNNs to other models.

WebQSP-WD We derive WebQSP-WD from the publicly available WebQSP dataset (Yih et al., 2016), which is a corrected version of the popular WebQuestions dataset (Berant et al., 2013). WebQSP contains natural language questions, Freebase IDs of the correct answers and SPARQL queries to retrieve them that also use the Freebase schema. Freebase was a common choice of a knowledge base at the time of the dataset creation, but was discontinued since then and is no longer up-to-date, including unavailability of APIs and new dumps. The Freebase entity linking API is no longer available that precludes us from using it for evaluation purposes. Most importantly, as we have argued in section 2.1, Wikidata offers a more modern and suitable alternative to Freebase for knowledge base question answering. Motivated by the advantages of Wikidata, we update the existing benchmark to be used with Wikidata and release it online under a public license[3].

[3] `https://github.com/UKPLab/coling2018-graph-neural-networks-question-answering/`

	train	test
questions	2880	1033
complex questions	419	293
avg. # of relations per question	1.35	1.39

Table 4.1: Dataset statistics for the WebQSP-WD, Wikidata knowledge base question answering benchmark.

```
{
  "answers": ["Q118771", "Q131792", "Q3783"],
  "answers_str": ["Rio Negro", "Orinoco", "Amazon River"],
  "questionid": "WebQTrn-246",
  "utterance": "what are the three major rivers in south
     america?"
}
```

Listing 4.1: A snippet of the WebQSP-WD question answering dataset in the JSON format. Here a single question is shown, we use 'answers' to store the Wikidata IDs of the correct entity set and 'answers_str' to store the original answer strings from WebQSP.

The questions in the WebQSP dataset were collected with the Google Suggest API and are thus more natural than manually constructed questions (cf. Free917 dataset in Berant et al., 2013). The answers were retrieved from Freebase with the help of crowdsourcing. Each answer is a non-empty set of entities. The dataset contains both simple questions that can be answered with a single relation and complex questions that require multiple relations and constraints. It is a common benchmark for semantic parsers and information retrieval systems and was used in the recent studies on knowledge base question answering (Reddy et al., 2017; Cheng et al., 2017; Luo et al., 2018). We map Freebase IDs in the WebQSP train and test partitions to Wikidata IDs using the manually specified Freebase ID for the Wikidata entries (see Section 2.1 for background). We keep the questions where the answer is a set of multiple entities, and we were able to map at least one of them. That means that we keep the question and the corresponding answer set even if we were not able to map all answer entities in the answer set to Wikidata. It is important to note that this does not ensure that a question is answerable with Wikidata as there still might be no relation paths in the knowledge base that connect the entities in the question with the answer. That makes the Wikidata setup more challenging, since the perfect result on this dataset is no longer possible. Yet, such dataset is sufficient for our purpose of benchmarking GGNN-based models against the standard architectures. We have successfully converted 3913 questions in WebQSP to Wikidata (Table 4.1). We designate this version of the dataset WebQSP-WD. The dataset is distributed in a JSON format and Listing 4.1 shows a snippet with a single question.

Metrics We follow the previous work on knowledge base question answering (Berant et al., 2013) and use precision, recall and F-score to evaluate the models in all of our experiments. The measures are computed for each individual question and then macro-averaged. This ensures a fair evaluation, since a system might provide a partially correct answer that is nevertheless better than a complete miss. For example for the question 'What are the Taylor Swift's albums since 2015', the system might fail to model the constraints properly and return simply a list of all Taylor Swift's albums. This would result in a perfect recall and low precision against the correct answer that is only a subset of her albums. However, it is still better than completely misinterpreting the question and returning a list of songs that results in zero recall and precision.

Yih et al. (2015, 2016) and Luo et al. (2018) describe the strongest results for knowledge base question answering with Freebase. The values reported on WebQSP-WD in this work are not directly comparable to the previous results for Freebase. To bridge our experiments with the prior work, we use a re-implementation of the Yih et al. (2015)'s STAGG model as one point of comparison for our models.

4.4 Representation Learning for Semantic Parsing

We follow the state-of-the-art approaches for question answering (Yih et al., 2015; Dong et al., 2015; Bao et al., 2016) and learn representations for the questions and the semantic graphs generated with the procedure described in Section 4.2. Having learned a model to embed the question and the semantic graph into the same multi-dimensional space allows us to compare questions and semantic graphs with each other using a similarity function. We use cosine similarity between the question representation and the semantic graph representation as a similarity function that judges whether a semantic graph is similar to a question: $\gamma = \cos(\mathbf{v_q}, \mathbf{v_g})$, where $\mathbf{v_q}$ is a vector encoding for the question and $\mathbf{v_g}$ is a vector encoding for the semantic graph. The semantic graph closest to the question in the multi-dimensional space according to the similarity function is taken as the semantic representation of the input question. In the following sections, we describe the architectures that we use to learn the representations for questions and graphs.

4.4.1 Encoding Questions into a Vector

We use iterated Dilated Convolutional Neural Networkss (DCNNs) to learn a representation for a question. CNNs have been proven effective for sentence processing on a variety of tasks, including NER (Strubell et al., 2017), relation extraction (Zeng et al., 2015) and knowledge base question answering (Dong et al., 2015). We have already successfully used DCNNs in our entity linking architecture in Section 3.2. A DCNNs architecture is depicted in Figure 4.6, where it is used to map an input question to a fixed-size vector representation.

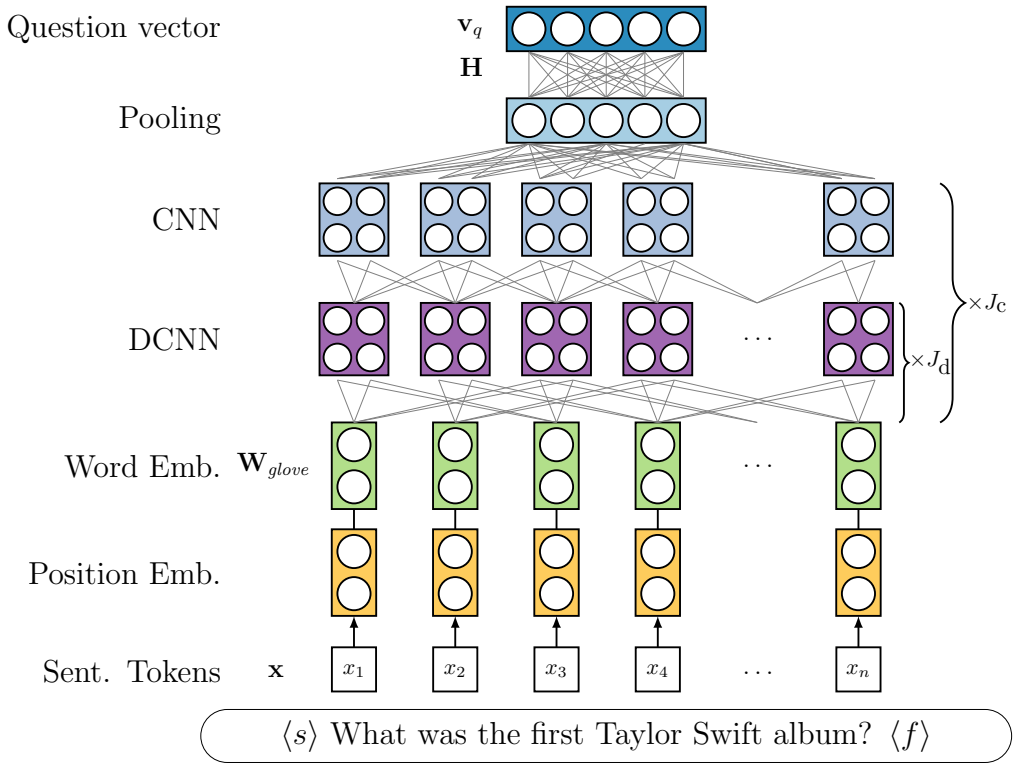

Figure 4.6: Iterated DCNNs architecture, here used to process an example question. The question is padded with starting and end tokens. We encode the tokens with pre-trained word embeddings and deterministic position embeddings. This input is processed by a block of DCNNs+CNN for J_c iterations. The output of the last iteration is pooled and transformed into the encoding for the question $\mathbf{v_q}$.

Our question encoder is similar to Zeng et al. (2015) except that we use dilated convolutions where they use simple convolutions.

The input question is first tokenized and the special start and end tokens are added to the sequence: $\mathbf{x} = \{\langle s \rangle, x_1, x_2 \dots x_n, \langle f \rangle\}$. Next, we map the tokens in $\mathbf{x}$ to d_w-dimensional pre-trained word embeddings, using a matrix $\mathbf{W}_{glove} \in \mathbb{R}^{|V_w| \times d_w}$, where $|V_w|$ is the size of the vocabulary. We use 100-dimensional GloVe embeddings that were trained on a 6 billion corpus (Pennington et al., 2014) and we do not update them during the subsequent training. To each word vector, we add a deterministic position encoding computed as in Vaswani et al. (2017).

We have also experimented with the ELMO contextualized word embeddings (Peters et al., 2018) in addition to GloVe. Those model versions have consistently underperformed on the development set and were thus not evaluated further. We tried to use the ELMO embeddings alone and in combination with GloVe, though to no improvement. ELMO has proven powerful for sequence labeling, but its effect on knowledge base question answering systems has not yet been uniformly studied. We hypothesize that for knowledge base question answering the identity relation between the same words in the question and the source is important. Partially opposed to that, the power of ELMO is to produce different

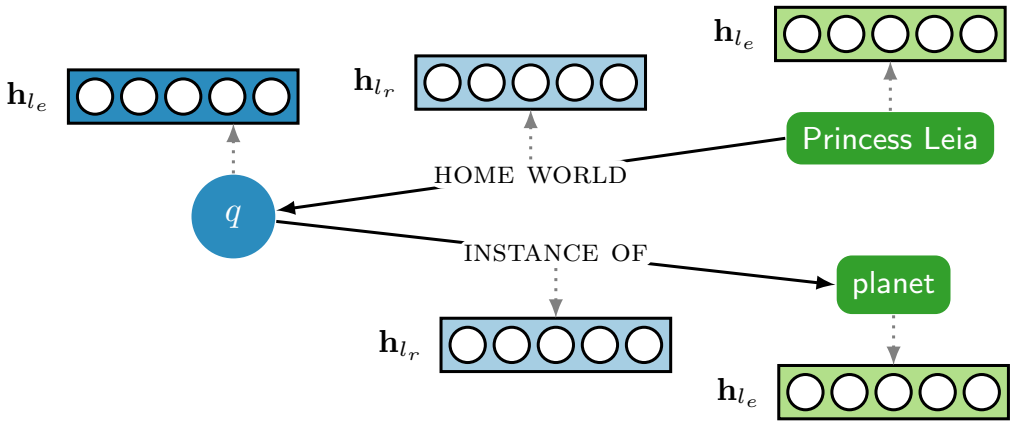

Figure 4.7: Encoding a semantic graph into initial hidden states for processing with a GNN. Each entity and relation is mapped to a vector representation $\mathbf{h}_{l_e}$ and $\mathbf{h}_{l_r}$. We use the entity and relation labels and pre-trained word embeddings to compute the vectors $\mathbf{h}_{l_e}$ and $\mathbf{h}_{l_r}$. The vector for the q variable is randomly initialized.

vectors for the same words depending on the context, which might be less crucial for question answering. The effect of contextualized word embeddings on knowledge base question answering requires further investigation and offers an exciting direction for future work.

The sequence of word embeddings is further processed by an array of DCNN layers (see Figure 4.6). We apply J_d DCNN layers with dilation increasing proportionally to the layer number (dilation = $2^j, j \leq J_d$), followed by a CNN layer without dilation (dilation = 1). The block of DCNNs+CNN is repeatedly applied J_c times (hence *iterated*) as in Strubell et al. (2017). We apply a pooling operation after the last CNN layer and transform the output with a fully connected layer $\mathbf{H}$ and a ReLU non-linearity. We take the resulting vector $\mathbf{v}_q$ as the representation of the question.

4.4.2 Encoding Graphs into a Vector

To learn the encoding for the semantic graph representation, we need an effective way of encoding a directed graph into a vector. We follow the recent work on embedding graphs for NLP (Marcheggiani and Titov, 2017; Schlichtkrull et al., 2018; Sun et al., 2018) and adapt Graph Neural Networks (GNNs) for our task. We introduced the GGNN architecture in Section 2.3 and we describe here how it can be used to learn vector representations in the context of question answering.

GNNs can process all information in the graph in a unified way into a single vector. GGNNs process graphs by iteratively updating representations of graph nodes based on the neighboring nodes and relations (Li et al., 2016). We adopt GGNNs for semantic parsing to learn a vector representation of a semantic graph. Li et al. (2016) gave a formulation of GGNNs for graphs that incorporates labeled nodes and typed directed edges, which we have described in Section 2.3. Each node in

their formulation is assigned a vector that can be initialized from any information about the node (e.g. a label if the node is a Wikidata entity). The edges are assigned matrices based on their type and direction. This assumes a small fixed amount of node and relation types, however it becomes impractical if the number of edge types is more than a dozen. In our case, we need to differentiate between hundreds of different knowledge base relations that can function as edges in our semantics graphs and at the same time keep a model that can learn from limited data and generalize (that is the number of learned model parameters should not be too large). It becomes computationally too expensive to learn a separate matrix for each relation type. We change Li et al. (2016)'s approach to incorporate labeled edges. To the best of our knowledge, we are the first to apply GGNNs to semantic parsing and knowledge base question answering.

We propose two ways to solve the problem of learning effective relation type representations. In both cases, we use labels from Wikidata to compute vectors for entities and relation types. This enables us to directly incorporate the information on millions of entities and hundreds of relation types from the knowledge base.

Relations as vectors instead of matrices The first proposal is to assign vectors to relations instead of matrices and use simple vector addition combined with a linear transformation to update the nodes.

For a graph $\mathcal{E}$, we extract a set of all entities $\mathcal{V}$ in the graph and a set of all relation types $\mathcal{R}$ of its edges. For an entity $e \in \mathcal{V}$ or a relation type $r \in \mathcal{R}$ we retrieve the label and tokenize it: $\mathbf{l} = \{l_1, l_2 \dots l_n\}$. Then we map each token in $\mathbf{l}_e, \mathbf{l}_r$ to a word embedding using the matrix $\mathbf{W}_{glove}$, sum them and process with a fully connected layer to get a single label vector: $\mathbf{h}_l = \tanh\left(\mathbf{W}_\mathbf{l}\left[\sum_{n=1}^{|\mathbf{l}|} w_n\right] + \mathbf{b}_\mathbf{l}\right)$. We initialize the hidden states for the graph nodes with the label vectors of the entities: $\mathbf{h}_v^{(1)} = \mathbf{h}_{l_e}$. We further transform the relation label vectors to get directional embeddings for relation types: $\mathbf{h}'_r = \mathbf{W}_{\rightarrow}\mathbf{h}_{l_r}$, $\mathbf{h}''_r = \mathbf{W}_{\leftarrow}\mathbf{h}_{l_r}$. The label vectors are used to initialize the hidden states for the graph nodes (Figure 4.7). Using the same word embeddings as an input to construct the question and relation representations has been shown successful in previous work (Jain, 2016).

With each node vector, we also concatenate an entity type or a relation type embedding vector: $\mathbf{h}_v^{(1)} = [\mathbf{h}_{l_e}, \mathbf{h}_e]$. These vectors are randomly initialized and updated during model training. They provide a way for the model to store information about particular relation or entity types.

This requires us to change the update equations of the GGNN recurrence as follows:

$$\mathbf{a}_v^{(t)} = \mathbf{A}_{v:}^{\top}\left[\mathbf{h}_1^{(t-1)\top} \dots \mathbf{h}_{|\mathcal{V}|}^{(t-1)\top}\right] + \mathbf{A'}_{r:}^{\top}\left[\mathbf{h}_1^{\prime\top} \dots \mathbf{h}_{|\mathcal{R}|}^{\prime\top}, \mathbf{h}_1^{\prime\prime\top} \dots \mathbf{h}_{|\mathcal{R}|}^{\prime\prime\top}\right] \tag{4.1}$$

$$\mathbf{z}_v^t = \sigma\left(\mathbf{W}^z\mathbf{a}_v^{(t)} + \mathbf{U}^z\mathbf{h}_v^{(t-1)} + \mathbf{b}^z\right) \tag{4.2}$$

$$\mathbf{r}_v^t = \sigma\left(\mathbf{W}^r\mathbf{a}_v^{(t)} + \mathbf{U}^r\mathbf{h}_v^{(t-1)} + \mathbf{b}^r\right) \tag{4.3}$$

$$\widetilde{\mathbf{h}_v^{(t)}} = \tanh\left(\mathbf{W}\mathbf{a}_v^{(t)} + \mathbf{U}\left(\mathbf{r}_v^t \odot \mathbf{h}_v^{(t-1)}\right) + \mathbf{b}\right) \tag{4.4}$$

$$\mathbf{h}_v^{(t)} = (1 - \mathbf{z}_v^t) \odot \mathbf{h}_v^{t-1} + \mathbf{z}_v^t \odot \widetilde{\mathbf{h}_v^{(t)}}, \tag{4.5}$$

where σ is the logistic sigmoid function and $\odot$ is the element-wise multiplication. The matrix $\mathbf{A} \in \mathbb{Z}^{|\mathcal{V}|\times 2|\mathcal{V}|}$ stores the structure of the graph: a row of the matrix $\mathbf{A}_{v:}$ records the edges between the node v and the other nodes in the graph (we differentiate between incoming and outgoing edges). The second matrix $\mathbf{A'} \in \mathbb{Z}^{|\mathcal{V}|\times 2|\mathcal{R}|}$ stores the relation types of the incoming and outgoing edges.

The main difference from the model defined in Li et al. (2016) that we have previously introduced in Section 2.3 is that we compute activations $\mathbf{a}_v^{(t)}$ based on both the node hidden vectors $\mathbf{h}_v^{(t-1)}$ (in green) and relation hidden vectors $\mathbf{h}'_r$ (in red in Eq 4.1). The $\mathbf{z}_v^t$ in Eq 4.2 and $\mathbf{r}_v^t$ in Eq 4.3 are update and reset gates, that incorporate information from nodes, relation types and from the previous step to update node hidden states at each iteration. We do not make updates to the hidden vectors of the relations and use them only to pass the information to the node hidden states.

This solution incorporates the relation type information as edges into the GGNN. However, it can limit the expressivity of the graph networks and the amount of interaction they can learn, since the relation representations are not updated during the recurrence. Such a model focuses on updating and learning entity representations and is limited with respect to processing of knowledge base relations which are arguably more important for finding the correct semantic graph.

Levi graph transformation with untyped relations Our second proposal is not to change the GGNN internal mechanism, but to update the structure of the input graph itself. We propose to use a more straightforward method to uniformly encode relations and entities in semantic graphs without limiting the GGNNs: the Levi graph transformation (Levi, 1942). This transformation allows us to put an equal focus on knowledge base entities and relations. The method was recently successfully applied on AMR graphs with a small number of relation types (Beck et al., 2018). We test it for a large-scale knowledge base graph here.

The transformation is performed as follows. We expand the constructed knowledge base semantic graph in such a way that we create a new node for each relation

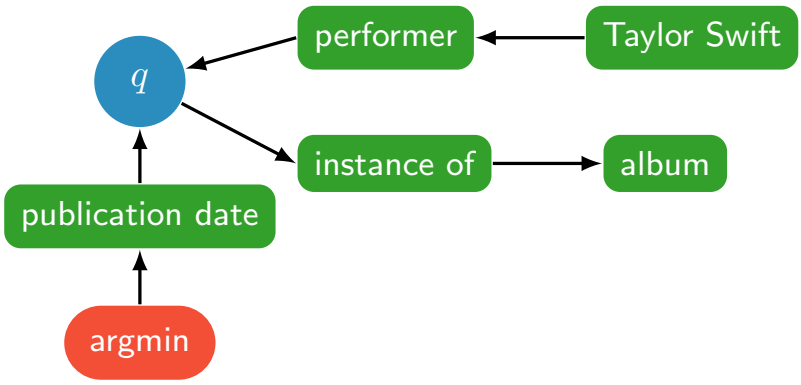

Figure 4.8: A graph for the example question "What was the first Taylor Swift album?" expanded with Levi graph transformation. The graph features two entities, a variable q and a constraint node argmin. The relations are represented as intermediate nodes between the entities they are connecting. The edges in the graph are not labeled but have a direction.

between the entities that it connects and label it with the title of the relations. Thus, the edges in the graph are no longer labeled and retain only the directionality (cf. Figure 4.4 and Figure 4.8). This allows us to have only two relation matrices that encode the direction of every edge, and treat the knowledge base relations as nodes that are being updated at each iteration. We use the entity and relation labels to initialize the node vectors in the same way as described above in the first proposal, and we randomly initialize the two relation matrices.

Li et al. (2016) gave a formulation of GGNNs for graphs with labeled nodes and typed directed edges. For this second variant, we adopt Li et al. (2016)'s formulation of GGNNs unchanged, as described in Section 2.3. We use it to process the expanded semantic graphs that consist of labeled nodes and *untyped* directed edges (treated as a single edge type). The model encodes nodes as vectors and requires only two separate relation matrices for updates along and against the direction of the edges (see Figure 4.9).

Graph vector For both variants, the GGNN recurrence is unrolled for a fixed number of steps T, and the gating mechanism works akin to Gated Recurrent Units (Cho et al., 2014). After the individual graph nodes and relations are encoded into the vectors, we apply the GGNN model on the set of node and relation embeddings and on the graph structure encoded as an adjacency matrix.

We unroll the recurrence of the GGNN for $T = 5$ steps in our experiments (Figure 2.16). It is recommended to use the number of steps that is at least as large as the longest path length over the graph Li et al. (2016). In initial experiments on the development set, we have tried out lower and higher number of steps, but it didn't result in any improvements. To produce the graph-level output vector, we take the hidden vector for the q-node at the last time step $t = T$ and transform it with a fully-connected layer and the ReLU non-linearity: $\mathbf{v}_g = \text{ReLU}\big(\mathbf{W}\mathbf{h}_q^{(t)} + \mathbf{b}\big)$. The

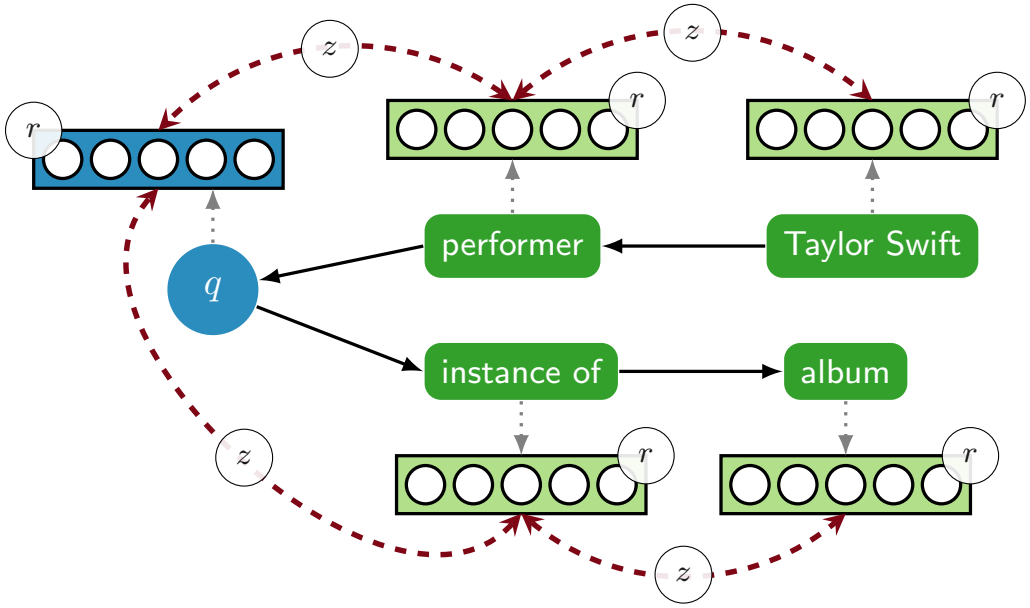

Figure 4.9: Processing an expanded semantic graph for a question with a GGNN. Each node standing for an entity, relation or the q variable is encoded as a vector. The dashed arrows show the updates along and against the direction of the edges. **z** and **r** are the update and reset gates.

graph-level output vector is taken as the representation of the semantic graphs. Hence, overall, we design a GGNN based encoder for semantic graphs that takes into account information about all entities and relations in the graph, including their labels, and about the structure of the graph and directionality of the edges. The model learns to summarize the most important aspects of this information in the graph output vector.

4.4.3 Model Architectures

We define six models for our question answering experiments, including three baselines and three graph models to evaluate our hypothesis that encoding semantic graph structure is beneficial for knowledge base semantic parsing.

1. STAGG (re-implementation of Yih et al. (2015)) — We implement a model that uses a combination of a neural network and manual features suggested in Yih et al. (2015) and in the follow up work of Bao et al. (2016). The approach of Yih et al. (2015) performed best among the previous work on complex questions (see Figure 4.3). First, a DCNN is used to produce a representation for the input question, as described in Section 4.4.1. Then, we replace the entity tokens in the input question with a special entity symbol $\langle e \rangle$ and apply a DCNN on it again to get a second representation for the question. For each semantic graph, we take the label of the relation in the first edge and take it to be the single core relation, as defined by Yih et al.

(2015) for their architecture. We tokenize the core relation label and likewise apply a DCNN on it to get a representation. We produce two representations for the core relation: one that includes the label of the attached entity and one that includes the entity symbol. The manual features include binary indicators for constraints in the semantic graphs and features for certain keywords in the question (see Yih et al. (2015) for a detailed description). In summary, this model only focuses on a single main relation in the graph and uses additional indicator features to model some of the graph properties.

The original system and the models of Yih et al. (2015) and of Bao et al. (2016) are not available and therefore we use our own implementation of their approach. There are small differences with the original model: we use DCNN instead of a simple CNN layer and train the DCNN together with the manual features, whereas Yih et al. (2015) pre-trained the CNN model on a separate corpus and used its output in the feature model.

2. Single Edge model — We use the DCNNs to encode the question and the label of the first edge of a semantic graph. The rest of the information is ignored. This model is similar to STAGG without manually added features.

3. Pooled Edges model — We use the DCNNs to encode the question and the label of each edge in the semantic graph. To get a fixed-vector representation of the graph, we apply a pooling operation over the representation of the individual edges. In that way, this model encodes all information about the semantic graphs, but disregards their structure. This model follows the approach of Luo et al. (2018).

4. Graph Neural Network (GNN) — To examine the effect of the gated graph neural architecture, we also include a model variant that does not use the gating mechanism and directly computes the hidden state as a combination of the activations (Eq 2.9) and the previous state.

5. Gated Graph Neural Network (GGNN) — We use the GGNN to process semantic representations (Section 4.4.2). This model encodes all information from semantic graphs, including their structure, into a vector representation. To encode the question, we use the same DCNN model as with the baseline models (see Section 4.4.1). In this version, we use labels to initialize relation vectors that are used to update the entity node. This the first variant of adopting GGNNs for knowledge base semantic graphs that we have described above in section 4.4.2

6. Gated Graph Neural Network on expanded graphs (ExpGGNN) — We label our final architecture ExpGGNN for *expanded graphs*. This is the second variant for incorporating the relation type information into the GGNNs. We encode the question with the same DCNN model as in other variants and apply the GGNN model on the expanded semantic candidate graphs.

The defined baselines use either manual features to capture the structure of the

semantic graph (STAGG), a simple pooling mechanism (Pooled Edges) or disregard the structure completely (Single Edge). The three graph models (denoted as GNN, GGNN and ExpGGNN) make the full use of the graph structure of a semantic representation and encode it with different variants of GNNs. With this setup, we are able to demonstrate what effect different levels of processing of the graph structure have on the final performance for knowledge base question answering. We do not include the published models of Berant et al. (2013) and Reddy et al. (2014) in the comparison since they were trained on Freebase and are not compatible with Wikidata. It is now not possible anymore to replicate exactly the studies that use Freebase due to the unavailability of the access APIs and new dumps[4]. We use the re-implementation of the more recent STAGG approach to put the graph models and our findings into the context of previous work and of feature-based models.

4.4.4 Model Training

To train the model, we need positive pairs of questions and semantic graphs. Since WebQSP does not contain annotated semantic representations for Wikidata, we use weak supervision, as suggested in Berant et al. (2013), to automatically produce training instances from question-answer pairs. Specifically, we follow Yih et al. (2015) and run our semantic graph construction procedure to create the training instances (see Section 4.2). We use the output of the S-MART entity linking system that is already available for the WebQSP dataset (Yang and Chang, 2015) to provide a set of entities $\mathcal{F}$ for each question.[5]

Once we generate the candidate semantic graphs for the given question and the extracted entity set with the procedure defined in Algorithm 1, instead of scoring the semantic graphs with the model, we evaluate each graph against Wikidata. We compare the answers extracted by evaluating the graph to the manual answers in the dataset for the given question. Semantic graphs that result in a correct answer are stored as positive training instances and the rest of the graphs generated during the same process are used as negative instances. Due to the aforementioned differences between Freebase and Wikidata (Section 4.3), we cannot retrieve the exact set of answers from Wikidata for some questions, which introduces additional noise into the procedure. For instance, Wikidata and Freebase might have slightly different list of main actors for a given movie. We filter out questions from the training set where we are not able to generate any positive training instances. We generate training examples for 1945 questions out of 2880 (see Table 4.1) and put 628 of them aside as a development set.

At each training epoch, we take all positive semantic graphs and up to 50 negative

[4] `https://developers.google.com/freebase/`

[5] S-MART itself is not openly available, but the output on the WebQuestions dataset was published by Yih et al. (2015): `https://github.com/scottyih/STAGG`

Parameter	Tested values	Selected
Hidden layer size	[32, 64, 128, 256, 512]	256
CNN filter size	[32, 64, 128, 256, 512]	256
CNN filter width	[3, 5, 7]	3
v_s, v_g sizes	[64, 128, 256, 512]	128
Dropout ratio	$\mathcal{U}(0.0, 0.5)$	0.2
DCNN	[1, 2, 3]	$J_d = 1, J_c = 1$
Pooling operation	[avg, sum, max]	max
Learning rate		$\alpha = 0.001$
Adam		$\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1e-08$

Table 4.2: Optimized hyper-parameter values for the ExpGGNN model in the experiments on the WebQSP-WD dataset.

graphs per question. We optimize the maximum margin loss function:

$$\mathcal{L} = \sum_{g \in C} \max\left(0, (m - \gamma(\mathbf{v}_q, \mathbf{v}_g^+) + \gamma(\mathbf{v}_q, \mathbf{v}_g^-))\right), \tag{4.6}$$

where C is a set of semantic graphs for the given question. From the loss function, we compute updates to the GGNN and the DCNN parts of the model and thereby learn similar vector encodings for questions and the corresponding correct knowledge base semantic graphs.

All models are trained using the Adam optimizer (Kingma and Ba, 2014) with a batch size of 64. We use an early stopping criterion on the development data to determine the number of training epochs. The learning rate is fixed to 0.001 and the other optimization parameters are set as recommended in Kingma and Ba (2014): $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1e-08$. We apply Dropout (Srivastava et al., 2014) at every fully-connected layer and on the embeddings layers. We determine the hyper-parameters, such as the size of the hidden layer, with the random search on the development set (see Table 4.2 for the tested range and the final selected value). On the 1945 training questions, the ExpGGNN model has usually finished training in under two hours on a single GPU.

4.4.5 Inference

We take the steps described in Section 4.2 to construct possible semantic graphs for a question at inference time. For WebQSP-WD, we use the entities produced by S-MART (Yang and Chang, 2015) to start the graph construction.

Each question and semantic graph are encoded into fixed-size vector representations, and the similarity function $\gamma = \cos(\mathbf{v_s}, \mathbf{v_g})$ is used to score the graphs. The highest scoring graph is used to retrieve the answers from Wikidata. Given the iterative nature of our semantic graph construction procedure, we adopt beam

search to speed up the computation. We score the graphs after each step and select the top 10 to proceed.

There are questions, which cannot be completely represented with the knowledge base schema, and only an incomplete graph may be generated for them. It is also the case that during the beam search we have to score and filter out incomplete graphs as well. Yet our model has seen complete semantic representations during training and might be biased to larger graphs even if a smaller graph is a better choice. We address this problem of processing the incomplete graphs with the size normalization trick: we normalize the scores computed by the model for each graph by the graph size.

Our idea is similar to length normalization for beam search, which is a common technique in machine translation (Yang et al., 2018). It compensates for the fact that the model was trained on full graphs but is being applied on incomplete semantic graphs during the beam search. During preliminary experiments, we have observed that an un-normalized model tends to prefer larger graphs, which it saw more during training. Size normalization also adds more flexibility, since the model might eventually select a semantic graph that does not incorporate all entities in the question. Such graphs would be less restrictive, and this strategy prioritizes recall.

4.4.6 Experimental Results

We compare the results on the WebQSP-WD test set (1033 test questions) in Table 4.3. As can be seen, the gated graph models outperform all other models across precision, recall and F-score, with the ExpGGNN model showing the best overall result. We confirm thereby that the architecture that encodes the structure of a semantic representation has an advantage over other approaches.

As described above, WebQSP-WD is a subset of WebQSP (Yih et al., 2016) that was originally built for Freebase. We have automatically mapped the Freebase IDs to Wikidata and the questions that do not have the mapping were filtered out. This still does not ensure that a question is answerable with Wikidata as there might be no correct relation paths in the knowledge base. The results are therefore lower in absolute values than those previously reported for Freebase and are not directly comparable with them. By benchmarking against the methods that use the same underlying training data and the knowledge base in a controlled experiment, we ensure a fair comparison.

We have repeated the experiments several times for the baseline models and for our new architectures, re-training the models each time with a different random seed. This allows us to explore how each architecture performs on average independent of the random initialization (Reimers and Gurevych, 2018). Table 4.3 shows the comparison of the question answering architectures including the standard deviation. The reported results are the average over 30 runs. We observe little variance in the results, the ExpGGNN model being the most stable.

	P	R	F	σ
STAGG (Y15)	0.1727	0.2140	0.1911	±0.120
Single Edge	0.2497	0.3073	0.2755	±0.018
Pooled Edges	0.2482	0.2925	0.2685	±0.022
GNN	0.2419	0.2890	0.2633	±0.020
GGNN	0.2589	0.3048	0.2799	±0.023
ExpGGNN	**0.3203**	**0.3752**	**0.3456**	±0.014

Table 4.3: Results for the graph neural models and the baselines on the WebQSP-WD test set, standard deviation σ is reported for the experiments that were repeated 30 times. Y15: Yih et al. (2015)

The STAGG architecture delivers the worst results in our experiments and has the largest standard deviation over the repeated runs. We suppose that the main reason is that STAGG relies on manually defined features to encode the graph structure, which are less comprehensive than GNN. The average over the top five runs for STAGG is 0.30, which is in line with the good single run results reported in Yih et al. (2015) on Freebase. The Single Edge model outperforms the more complex Pooled Edges model by a noticeable margin. Because the Single Edge baseline cannot differentiate between one-relation and multi-relational graphs, by the nature of the model, it prefers simple graphs that consist of a single edge. Always selecting a single edge graph is a good strategy to achieve higher recall values, since those are the least restrictive and there is less chance to choose a wrong relation if only one relation type is taken.

We can see in Table 4.3 that most graph models perform better than the baselines. The GNN model is the only exception. This model is on par with the non-graph models. Lacking the gating architecture, the GNN is less capable to learn interactions in the semantic graphs. Looking at the individual model predictions, we see that GNN more often predicts fewer relations that are needed and therefore gets only a partially correct answer. For example for the question "Who is the prime minister of Spain 2011?", GNN predicts a graph of two relations INSTANCE OF(human) and HEAD OF GOVERNMENT, which returns a list of all Spanish prime ministers. The complete correct graph would also include temporal constraints.

The ExpGGNN architecture proposed in this work outperforms the other variants and we thus prove that the Levi graph transformation in combination with GGNNs is the best strategy to encode structure for knowledge base semantic graphs. Since each architecture was trained multiple times to account for the effect of the random seed initialization, we can also inspect the distribution of the results across all runs. Figure 4.10 presents a detailed picture of the model's performance. We can observe that the results for the ExpGGNN are consistently better than for other models. Notably, the GGNN architecture that achieves the top second result on average demonstrates high variance across the repeated ex-

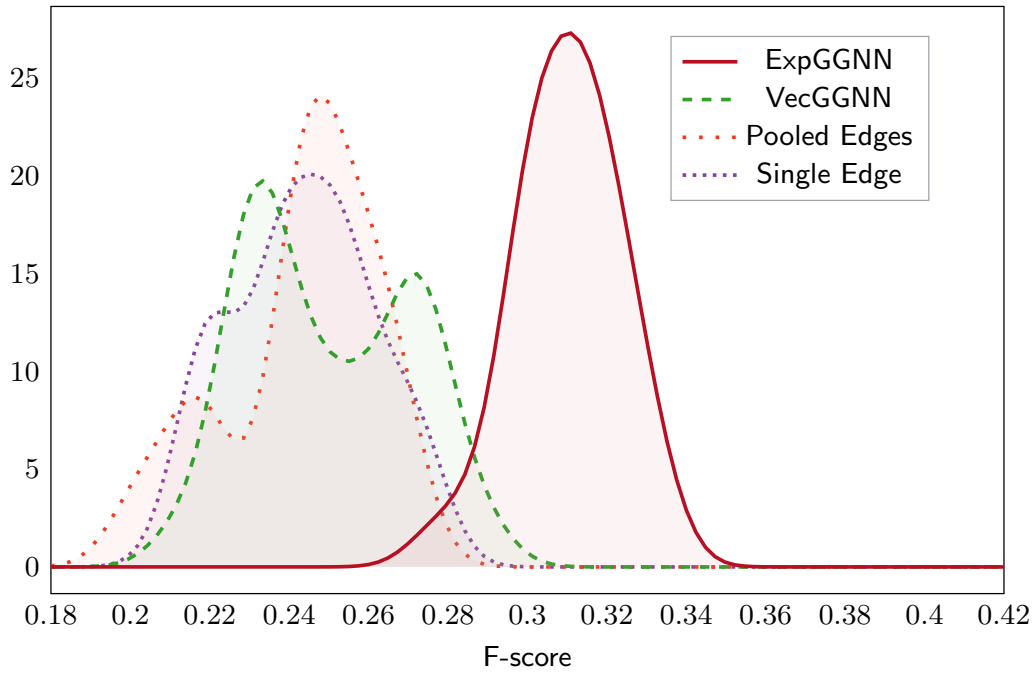

Figure 4.10: Distribution of the results for the graph neural models and the baselines on the WebQSP-WD test set across 30 experiment repetitions with different random seeds.

periment runs. The other models also show a long tail of lower performance cases.

Our main goal was to produce better encoding of complex semantic graphs and hence we break down the performance of the evaluated models by the number of relations that are needed to find the correct answer.[6] In Figure 4.11, we see that for the STAGG, Single Edge and Pooled Edges baselines, the performance on more complex questions drops dramatically compared to the results on simpler questions. The GNN and the GGNN models maintain a performance that is above the other baselines and they suffer a smaller drop on the most complex questions. However, the performance gap to the next baseline is small and the performance of the GGNN model drops for the most complex questions (number of edges ≥ 3) below of the GNN model. By looking at the model errors on the most complex questions, we could see that GGNN tends to incorrectly predict one of the relations in the graph, which results in a wrong answer.

We see that the ExpGGNN model offers the best results both on simple and complex questions, as it effectively encodes the structure of semantic graphs. It suffers a relative performance drop on the most complex questions but maintains a performance that is well above all other baselines and the competing graph-based models. Notably, ExpGGNN also has the best performance on the simplest semantic graphs that only have one edge. In these cases, two entities and one rela-

[6] We use the manually constructed queries provided by Yih et al. (2016) for WebQSP to estimate the number of relations needed to find the correct answer.

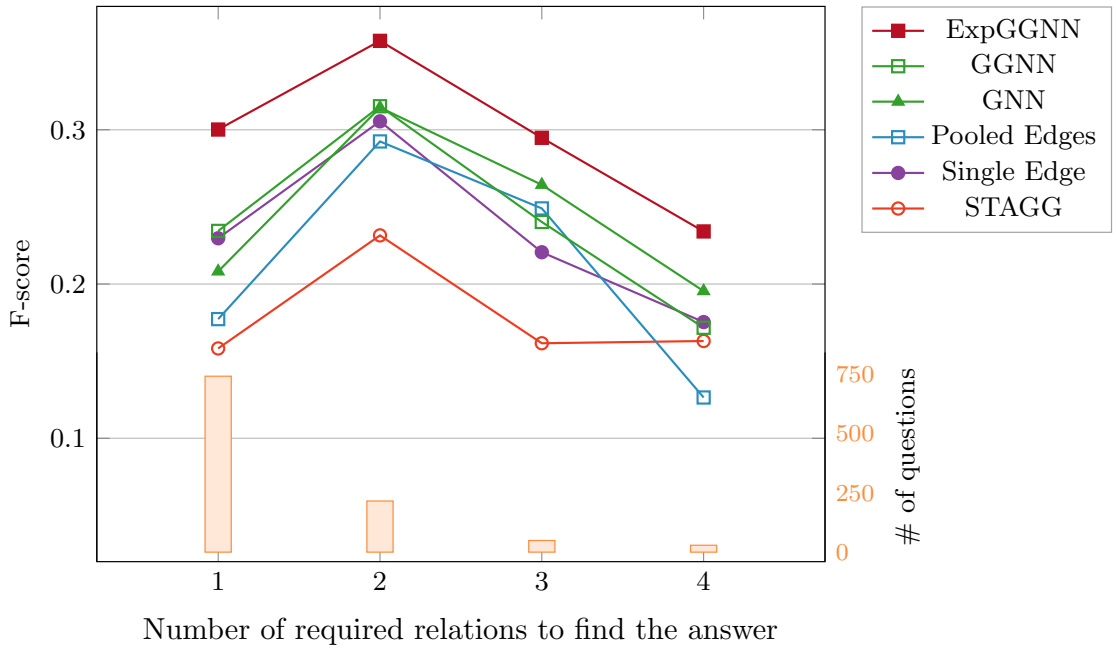

Figure 4.11: Evaluation results on the WebQSP-WD question answering test set by the number of relations per semantic graph needed to find the correct answer. The bars show the number of questions in the data in each category.

tion in the graph are expanded to a three-node graph and hence the components interact with each other through the two untyped edges in both directions. The ExpGGNN is better at capturing this interaction than other models.

We conclude from our experiments that it is important to model the graph structure and to put equal focus on relation and entity information in the graph. It is also advantageous to use a simpler neural architecture with less parameters overall: our best performing ExpGGNN architecture uses only a single relation matrix to model untyped edges in the expanded graph.

4.4.7 Error Analysis

To better understand the difference between the approaches that encode graph structure and the approaches that disregard it, we closely look at the output of ExpGGNN compared to the GGNN and Pooled Edges models. The first two rows in Table 4.4 show how often the respective model has returned an answer with an F-score higher than 0.5 and how often it returned an answer that was not just completely wrong (F-score $>$ 0.0). We see that ExpGGNN delivers an acceptable answer almost 41 % more often than the Pooled Edges model and 11 % more often than GGNN, but there is still a lot of questions that are not answered correctly.

We have manually analyzed 100 sample answers from the three models where the resulting F-score was lower than 0.5 (see Table 4.4). Since ExpGGNN makes fewer mistakes in general, dataset inconsistencies take a larger portion of the

	Pooled Edges	GGNN	ExpGGNN
Partial answers on the full test set			
F-score > 0.5	22.27 %	28.37 %	31.55 %
F-score > 0.0	29.82 %	37.08 %	45.98 %
hit@10	35.62 %	44.05 %	56.78 %
Manual error analysis on 100 sample questions			
Entity linker errors	6 %	8 %	4 %
No path to answer	14 %	18 %	10 %
Dataset inconsistency	8 %	14 %	22 %
Wrong relation or structure	72 %	60 %	64 %

Table 4.4: Error analysis of the model answers on the WebQSP-WD dataset.

final error count. In 10 % of the cases, it was impossible to find the answer in Wikidata since there was no path between entities in the question and the answer. For example, for the question "Where did Harper Lee attend high school?", the correct answer "Monroe County High School" is a valid entity in Wikidata, but it is not connected to "Harper Lee" via the EDUCATED AT relation. 22 % of the time, the dataset contained an inconsistent answer and even though the model has predicted the correct semantic graph, the answers did not match. For example, a correct answer for a question about someone's place of birth is usually a city or a town, yet for a smaller set of questions a city borough (e.g. Manhattan) or a country are listed in the dataset.

Overall, in 32 % of the cases the error was caused by the gap between the knowledge base and the dataset. We can use this to put the current results into a perspective with the previously reported numbers for the Freebase knowledge base. If we approximately adjust our results for this kind of errors, we achieve up to 0.56 F-score. Reddy et al. (2016) report results for multiple approaches ranging from 0.40 to 0.50 F-score on the original WebQuestions dataset and Yih et al. (2015) list 0.53 on WebQSP that is a superset of WebQSP-WD (see Section 4.3).

The majority of errors for both models are caused by a wrong prediction of the semantic graph (the last row in Table 4.4). ExpGGNN produces fewer wrong semantic graphs and more often successfully handles graphs with multiple edges. For example, for the question "What language do people speak in Brazil?", the ExpGGNN model correctly predicts the graph with two edges HOME COUNTRY and NATIVE LANGUAGE to get a list of all languages that are spoken in Brazil, whereas the baseline model selects the relation OFFICIAL LANGUAGE that returns only the official language of the country. We also look at the hit@10 measure that shows how often the correct semantic graph was in the top 10 scored graphs by the model (Table 4.4). Notably, in 56 % of the cases the correct semantic graph was still among the top scored graphs for the ExpGGNN model.

We only train on the WebQSP-WD dataset and we note that more data might be necessary to effectively train the gated graph architecture. Reddy et al. (2014) suggest an unsupervised learning method to learn a model from a large web corpus, while Su et al. (2016) use patterns and crowdsourcing to create new data with specific properties. These techniques can be used to further improve the performance of our model. Overall, we have demonstrated in the first part of this chapter that graph-based techniques can enable a much-improved knowledge base semantic parser for question answering. The most promising technique in our experiments was proven to be the one based on the Levi graph transformation and the GGNN architecture, the ExpGGNN model.

4.5 Application: Text Comprehension

Broadly, open-domain question answering can be divided into methods that retrieve answers from a structured knowledge base and those that search for an answer in an unstructured set of source documents. In the first half of this chapter (Section 4.1 to Section 4.4), we have regarded the task of finding an answer to the question in a knowledge base using grounded semantic parsing techniques. In this section, we examine how knowledge base semantic parsing can be directly applied as a method to add external knowledge to the document question answering task (also called text comprehension).

The document question answering methods rely on span selection methods and more recently have started to incorporate the external knowledge. However, they rarely make the interpretation of the input explicit. The knowledge base question answering systems, which we have discussed so far in this chapter, usually represent the question in a structured human-readable form. In this section, we venture into exploring the application of the knowledge base semantic parsing beyond only the knowledge base question answering data and apply it on a challenging open-domain document question answering dataset. This study is the first attempt to directly employ the grounded semantic parsing methods for document comprehension. We report empirical results for our system on the TriviaQA dataset. We select TriviaQA, because it is predominantly concerned with the world knowledge, but introduces a challenge that the questions are no longer guaranteed to have a Wikidata interpretation. Therefore, it is suited to evaluate the Wikidata-based architecture that we have used so far. After the results discussion, we charter future directions for this research, such as a tighter integration of semantic parsing with text comprehension models and techniques to extend the coverage of knowledge base semantic parsing.

Our flexible question answering architecture enables us to transfer the GNN model to the document question answering dataset and improve the results of a baseline architecture, as well as to add a layer of interpretability to it. Altogether, this yields an architecture that can dynamically use external knowledge to choose answers directly from the knowledge base or to filter out incorrect predictions. We use the

Question:
Which actor played Superman in Man of Steel?
Source Document:
Man of Steel is a 2013 superhero film [. . .] The film stars Henry Cavill, Amy Adams, Michael Shannon, Kevin Costner and Diane Lane. In 2013, director Zack Snyder rebooted the film franchise with Man of Steel, starring **Henry Cavill**. Cavill will reprise his role as Superman in the 2017 film Justice League.

Figure 4.12: An example question and a source document from the TriviaQA dataset. The correct answer is highlighted in bold, whereas the top choice of the span selection model without semantic parsing is underlined.

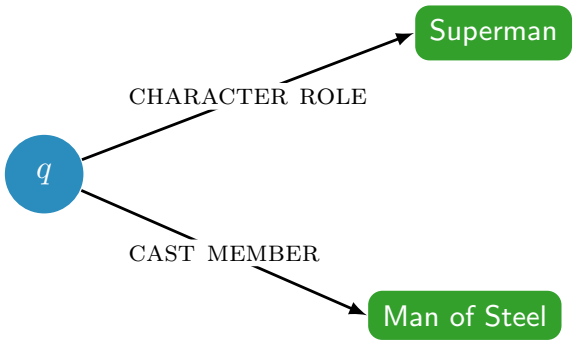

Figure 4.13: An automatically constructed knowledge base semantic graph that encodes the question from the TriviaQA dataset in Figure 4.12.

knowledge base semantic parsing model to interpret the question as a knowledge base semantic graph and instead of retrieving the answer from Wikidata, we apply it to filter answers from the output of a state-of-the-art document comprehension model.

4.5.1 Task Description

Open-domain question answering is an essential natural language understanding problem. Given a natural language question, the task is to find an answer in a trusted source of evidence (Liang, 2016; Rajpurkar et al., 2016). For example, we saw that for the question "What is the first Beatles album?" the answer, "Please Please Me", could be retrieved from a structured knowledge base, such as Wikidata. In the case of document question answering, the task is to find the answer in an unstructured text collection, such as Wikipedia or the web (see Figure 4.12).

Thereby, question answering approaches are broadly divided into those that consult structured information sources (the knowledge base question answering) and those that work with unstructured document collections (often named text comprehension or document question answering). The former methods are usually concerned with mapping the question to the structure of the knowledge base as we have extensively discussed in the previous section (Berant et al., 2013;

Bao et al., 2016; Reddy et al., 2017), whereas the latter rely on end-to-end neural networks conditioned on the question to select an answer span in the source documents (Hermann et al., 2015; Clark and Gardner, 2018). In this section, we combine the semantic parsing and the document question answering methods.

At the center of our approach is again the explicit modeling of the input using an open-domain knowledge base (see the graph in Figure 4.13). We select the TriviaQA document question answering dataset for evaluation, which is covered by Wikidata to the extent that allows us to verify our Wikidata-based system against the state of the art.

TriviaQA (Joshi et al., 2017) is a large open-domain document question answering dataset, where the system needs to find an answer for a trivia question. The dataset contains a set of supporting source documents and is divided into two parts based on the domain: *wiki* for Wikipedia articles and *web* for web documents. A crucial difference between TriviaQA and other contemporary document question answering datasets (e.g. SQuAD or NewsQA) is that TriviaQA questions were not crowed-sourced from a fixed paragraph but were authored independently, which makes them more challenging. The trivia domain is well covered by Wikidata, which makes it a fitting benchmark in our setting. The state of the art at the time of our experiments for the TriviaQA *wiki* dataset is 66 % against 79 % human performance and a 82.5 % upper bound (Joshi et al., 2017), which leaves an opportunity for further improvements.

4.5.2 Approaches to Text Comprehension

Given the abundance of source information in textual form, answering natural questions from documents has been at the core of NLP research for the last years (Hermann et al., 2015; Weissenborn et al., 2018b; Clark and Gardner, 2018). Multiple benchmarks were developed for different domains (Kaushik and Lipton, 2018), ranging from news (Hermann et al., 2015) to children books (Hill et al., 2016), to encyclopedia (Rajpurkar et al., 2016) and more.[7] The field is continuing to grow with the release of SQuAD 2.0 (Rajpurkar et al., 2018) and Natural Questions datasets (Kwiatkowski et al., 2019).

To test our hypothesis that a knowledge-based system can be used to enhance modern document comprehension architectures, we needed a dataset with sufficient Wikidata coverage. We select TriviaQA (Joshi et al., 2017) which contains trivia questions about common world knowledge, a domain reasonably aligned with Wikidata. The systems applied so far on TriviaQA have used an end-to-end span selection architecture (Joshi et al., 2017; Clark and Gardner, 2018; Wang et al., 2018; Weissenborn et al., 2018a). Joshi et al. (2017) note that 11 % of the errors are due to missing relevant relations between the entities in the question. For example, in the question in Figure 4.12 a system has to recognize that the correct

[7] We only cite a few exemplary papers here as the full overview of this field would be beyond the scope of our manuscript.

answer is a person who was a cast member of the movie "Man of Steel" and was playing the Superman. Failing to detect either of these relations results in an incorrect answer. Additional 18 % of errors on TriviaQA are caused by the relations in the evidence document that stretch over multiple non-adjacent sentences.

Weissenborn et al. (2018a) show that using relevant commonsense information from ConceptNet[8] and Wikipedia improves the performance of a system on TriviaQA. ConceptNet is a semi-structured resource akin to Wikidata but focusing primarily on commonsense and lexical knowledge. Weissenborn et al. (2018a) proceed by appending the commonsense information in a textualized form to the evidence documents. For example, the relevant knowledge triples from ConceptNet are retrieved with an information extraction method and using simple lexicalization rules the triples are translated into statements that are then appended to the documents. The integration of the Wikipedia knowledge is implemented similarly by appending relevant sentences from Wikipedia directly to the evidence documents. Weissenborn et al. (2018a) show that enriching the evidence with world knowledge by adding Wikipedia information improves the performance. However, this comes with additional noise as the model has more information in text form to process. The authors note that in 44 % of the cases the retrieved Wikipedia text does not contain useful information.

Moreover, adding the external knowledge in unstructured textual form does not improve the interpretability of model decisions. This is in contrast to the knowledge base question answering approaches that construct an explicit knowledge base semantic representation which encodes relations in the questions. Such representation can be easily inspected by humans (Figure 4.13). We explicitly model the question meaning as a semantic graph to include external knowledge into a document question answering system and to add to its interpretability.

4.5.3 Combining Text Comprehension and Semantic Parsing

We explore a straightforward late-fusion combination of text comprehension and knowledge base semantic parsing approaches. In this experiment, we apply the developed ExpGGNN model (Section 4.4) beyond just knowledge base question answering. Figure 4.14 shows a high-level overview of our system. We interpret the question with respect to Wikidata using the knowledge base semantic parsing system that we have successfully tested in Section 4.4.6. With the Wikidata semantic parsing we can construct a semantic graph that explicitly encodes the question (see the left box in Figure 4.14). We combine it with the state-of-the-art JackQA text comprehension model (Weissenborn et al., 2018b). We use the semantic parsing interpretation of the question to select an answer from the top candidates of the text comprehension model (the right box in Figure 4.14). Each candidate answer is put in place of the question variable and the validity of the resulting graph is checked with Wikidata. We do not apply the semantic parser

[8] http://conceptnet.io/

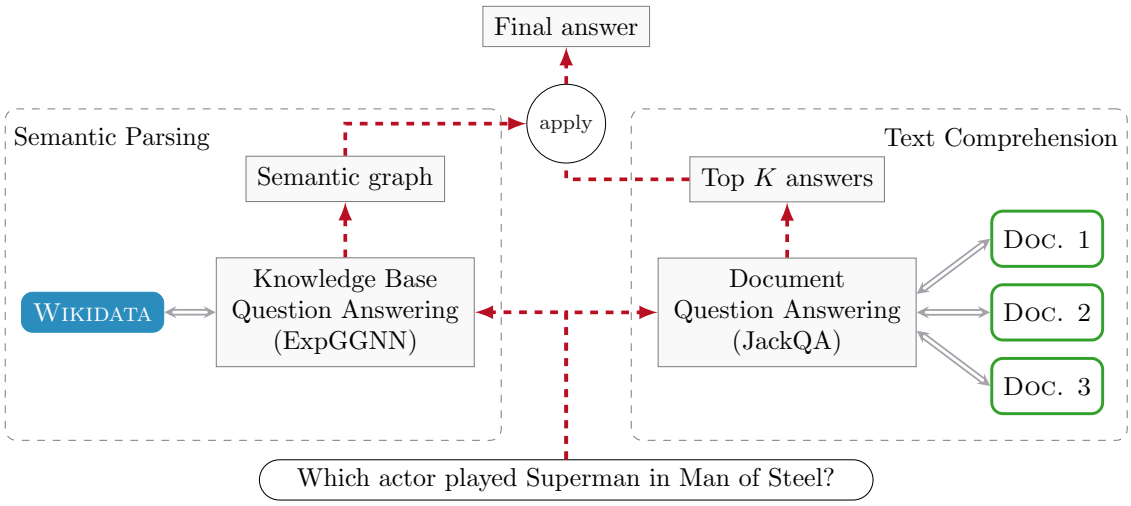

Figure 4.14: An overview of the proposed combination of a text comprehension model for document question answering and a semantic parsing model knowledge base question answering. The input question is processed by each model individually and the knowledge base interpretation is applied on the answers extracted from the source documents to filter them. The answer candidates of the text comprehension model that do not agree with the information in the knowledge base are removed.

interpretation if we are unable to construct a semantic graph for the input question with model confidence above a certain threshold.

For instance, for the question in Figure 4.12 the document question answering system first selects an incorrect span 'Kevin Costner' (underlined). After constructing a knowledge base semantic graph for the question (Figure 4.13), we can explicitly encode that the answer should be the performer of the role of Superman. We can check that relational constraint in the external knowledge base for the candidate answers from the text comprehension model and change the model output accordingly.

To construct a semantic graph for an input question, we use the same graph generation procedure as before (see Section 4.2). It does not rely on the syntax of the question and allows to construct flexible graphs. This is crucial for covering real world datasets such as TriviaQA. We do not require the semantic parser to construct a full graph representation of the question meaning. Even with a partial semantic representation the knowledge base semantic graph may capture important constraints on the answer.

For each input question, we construct a set of possible graphs and use the same representation learning framework as in Section 4.4 that relies on DCNNs and GGNNs to encode graphs and questions into vectors. We use the best performing technique, the ExpGGNN model, which expands the graph relations into nodes and processes the resulting structure with GGNNs. The question encoding is performed with dilated convolutional networks (DCNNs). Similarly, we use the

cosine similarity function $\gamma = \cos(\mathbf{v_s}, \mathbf{v_g})$ to judge if a candidate knowledge base semantic graph matches the input question. We take the most similar graph to represent the question.

World knowledge relations are hard to learn from the distributional information only (Petroni et al., 2019) and hence, a text comprehension model might struggle to capture them. Our experiment shows that a knowledge base semantic parser can be directly transferred to the document question answering task at almost no additional implementational cost. We are the first to demonstrate that explicit modeling of meaning with a knowledge base can be used as a direct extension of a document question answering system.

4.5.4 Experimental Results

We present empirical results on the challenging TriviaQA benchmark, using both the *wiki* and *web* partitions of the dataset (Joshi et al., 2017). To start the graph generation procedure on the *wiki* partition, we map Wikipedia document names that are supplied with each example to Wikidata entities. On the *web* part, we rely on our entity linker from Section 3.2 to extract the set of entities in the input question.

We repeat the weak training data generation steps from Section 4.4 on TriviaQA to create training data. We use the questions from the TriviaQA training splits (61K questions for the *wiki* part and 76K for the *web* part of the dataset) to generate weakly supervised semantic graphs. For each question, we generate all possible semantic graphs that can be used to retrieve the correct answer and up to 100 negative graphs.

We use the same loss function as before (maximum margin loss, Equation 4.6) to train the model and take all positive graphs and sample 50 negative graphs per question during training. We use the Adam optimizer (Kingma and Ba, 2014) and an early stopping criterion on the TriviaQA development data. The learning rate is fixed to 0.001 and the other optimization parameters are set as recommended in Kingma and Ba (2014). We apply Dropout (Srivastava et al., 2014) after the pooling layer of the DCNN architecture and on the graph vector of ExpGGNN. We re-use the same values of the hyper-parameters as in the previous ExpGGNN experiments (Table 4.2). The model is re-trained five times with different random seed initializations to get the results on the development set. To get the results on the test set, we submit the output of a single model trained on the combined train and development set to the official TriviaQA evaluation web site. The training and evaluation is separate for *wiki* and *web* partitions.

During inference, we follow the same steps as in Section 4.4. Each question and semantic graph are encoded into vectors and the similarity function γ is used to rank the graphs. Given the iterative nature of the graph construction, we adopt beam search to speed up the computation and normalize the scores by the graph size. The procedure for the ExpGGNN model is essentially the same

as in the experiments in Section 4.4.5. We know that Wikidata does not have a perfect coverage of the TriviaQA *wiki* and *web* partitions. Not all questions in the dataset can be modelled with the Wikidata knowledge base schema. Hence, the main difference to the inference procedure in Section 4.4.5 is that we use a threshold on the ExpGGNN similarity score (the output of γ) to discard the semantic graph interpretation when the knowledge base semantic parsing model is uncertain.

We report the results for the transfer of the knowledge base semantic parsing system on the TriviaQA benchmark and show that it is able to improve upon the end-to-end text comprehension architecture for span selection. Our intention is to show the value that a structured knowledge base can bring for open-domain document question answering. We employ JackQA (Weissenborn et al., 2018b) as a basis for our combined architecture. JackQA is a freely available pre-trained state-of-the-art document question answering system with an extendable design. We process each question with JackQA and store the top K candidate answers. We use the ExpGGNN highest scored knowledge base semantic graph for a question to select an answer from the stored top candidates. We progress downwards through the candidates and return the first answer that can be put in place of the question node so that all relations in the semantic graph hold true according the information in Wikidata. We use the output of the JackQA directly if the knowledge base semantic parser model cannot construct a valid semantic graph due to information missing in the knowledge base.

Such late fusion of the two techniques is straightforward and allows us evaluate the effect of each subsystem on the overall performance. For the questions that were processed by the knowledge base semantic parsing system, we can also inspect the generated graphs, which adds to the interpretability of the system.

Table 4.5 and Table 4.6 show results on the TriviaQA dataset for the vanilla JackQA and our combined system. The TriviaQA *wiki* partition contains 7993 (development) and 7701 (test) questions, the *web* partition contains 9951 (development) and 9509 (test) questions. We report exact match results (EM) between the answer in the dataset and the produced answer set. We also include the F-score computed on the correct answer set and the produced answer the same way as introduced in Section 4.3. TriviaQA contains distantly supervised and manually verified test sets for each partition. The verified part contains a small amount of the original test set but was manually checked by human annotators. We refer to Joshi et al. (2017) for further information on the annotation of the dataset.

We observe that the explicit modeling of the question with knowledge base relations in our system increases the performance both in the exact match and in the F1 evaluation. The performance increase is consistent across the *wiki* and *web* partitions and across the distant and verified parts. The difference 0.002–0.009 is comparable to the previously reported performance margins on the TriviaQA dataset (for comparison Weissenborn et al. (2018a) report 0.006–0.01 improve-

		Distant		Verified	
		EM	F1	EM	F1
wiki	JackQA	0.5944	0.6488	0.6465	0.7109
	+ExpGGNN	**0.5981**	**0.6511**	**0.6667**	**0.7275**
web	JackQA	0.6064	0.6515	0.6808	0.7225
	+ExpGGNN	**0.6092**	**0.6539**	**0.6849**	**0.7257**

Table 4.5: Evaluation results on the TriviaQA development partitions.

		Distant		Verified	
		EM	F1	EM	F1
wiki	JackQA	0.5894	0.6465	0.6455	0.7021
	+ExpGGNN	**0.5924**	**0.6481**	**0.6541**	**0.7067**
web	JackQA	0.6064	0.6623	0.7334	0.7788
	+ExpGGNN	**0.6095**	**0.6649**	**0.7347**	**0.7793**

Table 4.6: Evaluation results on the TriviaQA test partitions.

ment for their model) and constitutes a significant improvement[9]. The standard deviation across different random initializations on the development set is only $\sigma = 0.00099$ for the exact match and $\sigma = 0.00074$ for F-score. Our semantic parsing architecture is flexible so that it can be combined with other document comprehension models, as well.

4.5.5 Error Analysis

Since our system constructs explicit knowledge base semantic graphs to represent the question meaning, we can manually inspect them. We looked at the cases when the integration of the grounded semantic parser results in the corrected answer compared to the JackQA baseline. In 80 % of the corrected cases, the change is caused by a correctly identified semantic graph representation. In 20 % of the cases, the semantic parsing component causes the system to change the answer to the correct one but the constructed graph representation is semantically wrong.

Some questions are not fully covered by Wikidata or the meaning of the question cannot be represented using the available knowledge base information. On the TriviaQA training set, our semantic graphs can successfully model 43 % of the data. For example, the question "The initial slogan for the American War of Independence was 'No taxation without' what ?" asks to complete the phrase.

[9] The difference between the JackQA and the combined model with ExpGGNN is statistically significant according to the Wilcoxon rank-sum test applied on the F-score results of the different model initializations.

We cannot model this question within the existing Wikidata schema.

Some limitations of the system become evident during the error analysis and present an opportunity for improvement in the future work. As noted above, Wikidata scheme does not cover some of the input question and the semantic graph interpretation is not possible. When our knowledge base semantic parser tries to predict a semantic graph in such cases, it leads to a nonsense output. We relied on the model confidence threshold to filter some of the noisy semantic representations. However, the semantic parser confidence is not a reliable factor to decide if the input can be represented with a knowledge base schema. Training the semantic parsing system to know when the input cannot be represented (e.g. to output an empty semantic graph if there is no good knowledge base representation) might reduce the noise further.

In this experiment, we have only explored the first step to a combined text comprehension and semantic parsing system with late-fusion of the components. An early-fusion architecture that learns to dynamically choose between different information sources would be an exciting future research direction and might result in further improvements. In the context of this dissertation, the conducted text comprehension experiment shows that the developed grounded semantic parsing method is directly applicable beyond just the datasets that were specifically created for knowledge base semantic parsing.

4.6 Sequence-to-sequence Semantic Parsing

In the previous sections of this chapter, we have explored and improved the semantic parsing approach for knowledge base question answering. We used step-by-step semantic graph generation and GGNNs to construct a robust architecture for semantic parsing. For the question answering task, the semantic graph produced by the parser is converted to a knowledge base SPARQL query to retrieve the answer to a question. In this section, we report on a collaboration project that investigates if an alternative sequence-to-sequence approach to mapping a question to a SPARQL query would be possible.

Most knowledge base question answering and semantic parsing systems, including the main approach in this thesis, construct a pipeline of entity linking, relation extraction and graph construction. We investigate if it is possible to bypass these steps with a single end-to-end model to decrease the error propagation. We adopt the sequence-to-sequence framework and use a large amount of synthetically generated data to train a Transformer model (Vaswani et al., 2017) with enough coverage for an open-domain question answering system. In the analysis, we show that a sequence-to-sequence model can suffer from incorrectly matching the knowledge base schema, but can improve the speed and precision of a question answering system. Application of fully end-to-end methods is particularly challenging for complex question that require multiple knowledge base relations. Overall, a sequence-to-sequence model achieves low recall performance on its

own. We combine the Transformer model with the semantic parsing architecture from Section 4.4 to show how the strengths of the two approaches can be used together. The combined architecture improves the result by 2 pp. F-score over the previously introduced ExpGGNN semantic parsing architecture. This shows that there is added value in the sequence-to-sequence approach. Our collaboration in Sorokin et al. (2019)[10] is foundational to this section. We use the `tensor2tensor` implementation[11] to create the sequence-to-sequence models.

4.6.1 Translating a Question into an Executable Query

So far, we have constructed semantic graphs as semantic representations and deterministically translated them into knowledge base queries, which were evaluated to retrieve the answer (see Section 4.1). In the case of open-domain knowledge bases, the common choice is the SPARQL query language that specifies a query over an RDF triple store. A SPARQL query encodes structured information similarly to a semantic graph and hence a semantic parsing problem can be framed as mapping an input question to a SPARQL query directly (cf. Figure 2.11).

Figure 4.15 compares a typical semantic parsing for question answering pipeline and a sequence-to-sequence architecture. A semantic parsing pipeline for question answering has several steps such as entity linking, relation extraction and graph construction. The constructed graph is then converted to a SPARQL query. The advantage of this setup is that each step limits the search space for the next one. For instance, selecting the entities in the question restricts what types of relations are possible. This ensures that the final query conforms to the knowledge base and that a non-empty answer is always returned. However, the multiple pipeline steps can collectively lead to error propagation. For instance, Yih et al. (2015) point out that incorrect entity linking accounts for more than 10 % of false predictions. Additionally, the execution of multiple steps, which require repeated communication with the knowledge base, affects the overall runtime of the system. An entity linking system has to retrieve knowledge base information for each candidate entity and the semantic graph construction checks for the available relations in the knowledge base at each generation step.

On the contrary, sequence-to-sequence mapping architectures have offered a complete end-to-end alternative for some of the structured prediction tasks (e.g. constituency parsing in Vinyals et al., 2015). Little effort was devoted so far to applying these methods in the knowledge base question answering setting. Training a sequence-to-sequence model from scratch is data hungry as the model needs to learn both the SPARQL syntax and the vocabulary of thousands of relations and entities. Thus, the main limiting factor for the sequence-to-sequence modelling is the small amount of training data available for knowledge base question answer-

[10] My contributions in this work are the following: research project lead, experiment planning and result analysis, application of the semantic parsing system on the task data and combination of sequence-to-sequence and semantic parsing systems.

[11] `https://github.com/tensorflow/tensor2tensor`

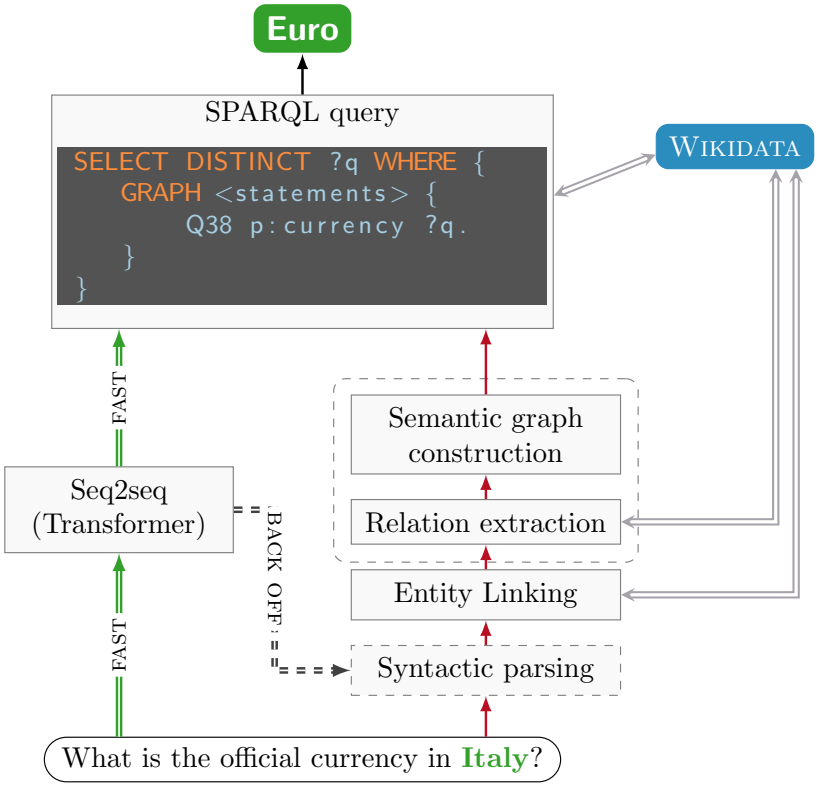

Figure 4.15: Semantic parsing and sequence-to-sequence approaches to knowledge base question answering. A knowledge base question answering pipeline (the red arrows on the right side) consists of multiple steps and requires repeated calls to Wikidata. Syntactic parsing is not used in our ExpGGNN semantic parser and the relation extraction and graph construction are unified in a single step, but two separate steps and repeated communication with the knowledge base are still required. A sequence-to-sequence architecture (the green arrows on the left side) encapsulates the processing of the question into a single step, but may result in an incoherent output that violates the information in the knowledge base. It is possible to use the semantic parser as a back-off method for the sequence-to-sequence model.

ing. We attempt to bridge this gap by relying on a question generation method to create a sufficient amount of synthetic training data and by training a state-of-the-art Transformer architecture (Vaswani et al., 2017) for sequence-to-sequence mapping.

A sequence-to-sequence architecture does not have an access to the knowledge base information during the query generation and therefore it might generate SPARQL queries that violate the knowledge base schema. For that reason, we explore a combination of a Transformer model and the previously developed semantic parsing pipeline. We propose a combined system where a high precision Transformer model is applied first and if it does not return an answer, a semantic parser is used (see the 'back off' in Figure 4.15).

4.6.2 Combining Shallow and Deep Semantics

In this experiment, we demonstrate that an end-to-end sequence-to-sequence architecture (we use a Transformer variant) can be successfully applied to open-domain knowledge base question answering. To overcome the lack of available training data, we automatically generate training questions and transfer the resulting model to real open-domain question answering data from the WebQSP-WD dataset (see Section 4.3).

End-to-end sequence-to-sequence models were first developed for machine translation (Sutskever et al., 2014). Several improvements were made to handle rare words or named entities that are the same in different languages (Luong et al., 2015). The idea to use sequence-to-sequence also for structured output is foundational for our work. Vinyals et al. (2015) were the first to employ this framework to present syntactic parsing as a type of a sequence-to-sequence problem. They have demonstrated promising results despite the fact that a syntactic parse is a tree and not a sequence. Dong and Lapata (2016) have experimented with sequence-to-sequence models for semantic parsing. Their model is trained to translate questions into sequences of actions. They only experimented on closed-domain datasets and did not integrate entity linking into the model.

Our evaluation setup is also similar to the task of translating natural language queries into the SQL language (Yu et al., 2018). However, the application domain of SQL is more limited in the number of entities and relations compared to Wikidata SPARQL and thus, SQL generation does not pose a similar challenge as the task that we consider in this section.

Synthetic training data In our experiments, we use the Wikidata knowledge base. Sequence-to-sequence models and even more so the modern Transformer architecture require a substantial amount of training data (Sutskever et al., 2014; Devlin et al., 2019b). To map questions to knowledge base queries, we need a dataset of questions paired with corresponding correct queries.

The largest available resources that contain SPARQL queries are the crowd-sourced WebQSP (Yih et al., 2016) and GraphQuestions (Su et al., 2016) datasets of a little less than 6000 questions each. Both datasets include SPARQL queries of different complexity that might use two or more semantic relations. The SimpleQuestion (Bordes et al., 2015) dataset contains question-relation pairs but is limited to questions about simple single triple assertions only.

The aforementioned datasets employ the schema of the Freebase knowledge base which was discontinued in 2016. It is not trivial to convert Freebase relation types to other knowledge bases, since there is no one-to-one correspondence (Azmy et al., 2018). We use the WebQSP-WD dataset for the model evaluation that was introduced in Section 4.3 and that is sufficiently covered by Wikidata. We do not use GraphQuestions for evaluation due to a low Wikidata coverage on the dataset (only 20 % of GraphQuestions answers are in Wikidata).

Template	SPARQL	Example
Who is the mayor of ⟨*b*⟩?	SELECT DISTINCT ?e WHERE { <b> p:P6 ?e }	Who is the mayor of London?
How many ⟨*c*⟩ does ⟨*b*⟩ have?	SELECT (COUNT (*) AS ?c) WHERE { ?e p:P31 <c> . ?e p:P17 <b> }	How many lighthouses does Denmark have?
Where was the youngest child of ⟨*c*⟩ born?	SELECT (DISTINCT ?e) WHERE { ?b p:P569 ?c . <c> p:P40 ?b . ?b p:P19 ?e } ORDER BY ?c LIMIT 1	Where was the youngest child of Prince Charles born?

Table 4.7: Examples of the defined templates for question generation and the corresponding SPARQL queries. The templates use variables ⟨*a*⟩, ⟨*b*⟩, ⟨*c*⟩ that are replaced with Wikidata entities in the instantiations of the templates.

To train the end-to-end models, we compose a large mixed training dataset from manually annotated Wikidata resources and automatically generated questions. We start with two small online resources that contain questions coupled with Wikidata SPARQL queries: a set of example queries for the Wikidata online endpoint[12] and the WDAquaCore0Questions dataset released by the WDAqua web service (Diefenbach et al., 2017). Both sets of questions are diverse with respect to the question domain and contain infrequent SPARQL operations, like UNION, BOUND and MINUS.

We manually define templates to extend our dataset. We adopt the template-based question-query generation procedure of Su et al. (2016), which targets different knowledge base relations and types of constraints. We inspect the questions in the SimpleQuestions and GraphQuestions datasets and use them to create the templates. We diversify the set of templates further by manually paraphrasing them. A template is a question with placeholders for entity mentions and a SPARQL query with corresponding placeholders for entity IDs. Table 4.7 shows examples of the defined templates. We define 292 templates for a wide range of topics and question types in total.

To generate question-query pairs from a template, we query Wikidata for the combinations of entity IDs that fit the SPARQL query template. That is, if we put the retrieved entity IDs into the SPARQL query instead of the placeholders, the resulting SPARQL query would produce a result when evaluated. For the retrieved entity IDs, we extract the Wikidata labels and put them instead of the placeholders into the question template.

To ensure more lexical variation in the training set, we use the state-of-the-art zero-shot question generation method of Elsahar et al. (2018). The authors train an encoder-decoder model to generate naturally sounding questions from knowledge

[12] https://query.wikidata.org

Source	Size	Avg. question length	Example
1. WDAquaCore0Questions	408	3.3	captain of the german national team?
2. Wikidata example queries	369	6.7	cities connected by the Trans-Siberian Railway?
3. Templates	103,996	8.1	how many people live in usa?
4. Neural generated	7700	7.5	what language is spoken in the film road house?
1. + 2. + 3. (T)	104,773	-	-
1. + 2. + 3. + 4. (T + G)	112,473	-	-

Table 4.8: Partitions of the created synthetic training SPARQL question answering dataset, which includes generated questions and corresponding SPARQL queries. (T) stands for questions generated from templates plus WDAquaCore0Questions and the example queries from the Wikidata endpoint. (G) stands for question generated with a neural model.

base triples. We select random Wikidata triples with the same relation as in the manually defined templates and generate from them additional 7700 question-query pairs with the model from Elsahar et al. (2018). Su et al. (2016) have also asked human annotators to write paraphrases for the automatically generated questions. This step is time-consuming and costly. Since we only need data for training, we do not employ human annotators and limit our training dataset to automatically generated questions.

By combining the template-based questions, the automatically zero-shot generated data, the Wikidata example queries and WDAquaCore0Questions, we create an open-domain synthetic dataset of 112,473 question-query pairs. It contains both simple question that require a single Wikidata relation to find an answer and complex questions that can be answered by combining several relations. We use the constructed examples for training a sequence-to-sequence model. Table 4.8 presents an overview over the created training set partitions. We evaluate two combinations of the training datasets: the one that uses primarily the template generated data (T) and the one that adds questions generated with the Elsahar et al. (2018)'s neural model (T+G).

Model Architecture We rely on the state-of-the-art Transformer architecture of Vaswani et al. (2017). Originally developed for the machine translation task, this architecture is now widely used for other sequence transduction tasks (Sutskever et al., 2014) and for pre-training universal sentence and word encoders (Devlin et al., 2019b). For comparison, we also include an LSTM with attention architec-

ture (Luong et al., 2015), which was widely used for sequence-to-sequence tasks before Transformers.

Our Transformer and LSTM architectures follow the same encoder-decoder structure. The main difference between a recurrent model, such as LSTM, and a Transformer is that an LSTM goes over the input sequentially and uses attention as an additional mechanism for long-range dependencies, whereas a Transformer processes all tokens in parallel and relies only on the attention to learn the dependencies. The core component of the Transformer is multi-head attention: instead of only performing a single attention function, it computes multiple attention weighted sums over the input (see Vaswani et al., 2017 for a more detailed description and motivation of Transformers).

For all models, we determine the number of training epochs on the validation part of the synthetic training dataset. The inner dimension of our model is $d_{model} = 512$. We use 8 attention heads with dimension of 64 for each head in the Transformer model. For regularization, we apply a 0.1 dropout to the output of each sub-layer and on the sum of the embeddings and positional encodings. We also use a label smoothing of 0.1 and a decaying learning rate. We keep the rest of the model parameters the same as in the original `tensor2tensor` implementation[13].

We use beam search with the beam size of 10 to generate the SPARQL query at test time. We take the highest-ranking output query and evaluate it against the Wikidata SPARQL endpoint to retrieve the answer. We have also experimented with using up to top 10 generated outputs and progressing downwards through them in case the top one results in an empty answer. This strategy did not improve the recall but hurt the precision of the model.

4.6.3 Experimental Results

We adopt two evaluation strategies: (i) Following the previous work on sequence-to-sequence semantic parsing (Soru et al., 2017), we compare the output to gold SPARQL queries (BLEU scores). (ii) We execute the queries against the knowledge base and compare the retrieved answers to gold answers in the dataset (query execution results). The second setup is more challenging as it requires a fully syntactically and semantically correct query and represents a real-life question answering task. We use macro precision, recall and F-score to compare the models.

BLEU score results To validate the usage of the Transformer architecture for this task, we first compare it to the only other previously published result on directly mapping questions to SPARQL (Soru et al., 2017). Soru et al. (2017) use an RNN-based sequence-to-sequence model and evaluate only using the BLEU score without executing the generated queries. The BLEU score is a common metric

[13] `https://github.com/tensorflow/tensor2tensor/blob/master/tensor2tensor/layers/common_hparams.py`

	BLEU
Soru et al. (2017)	0.753
LSTM+attention	0.796
Transformer	**0.945**

Table 4.9: Query comparison results on the Monuments300 SPARQL generation benchmark for the sequence-to-sequence models.

	BLEU
LSTM+attention	0.886
Transformer	**0.966**

Table 4.10: Query comparison results on the template partition (T) of the synthetic SPARQL generation dataset.

for evaluating machine translation results which estimates an n-gram overlap between two sequences. For direct comparison, we follow the evaluation setup of Soru et al. (2017) and train and evaluate on the Monuments300 dataset, which they use in their work. It contains 8344 generated closed-domain questions with DBpedia SPARQL queries. The Transformer architecture outperforms the RNN-based models by a large margin (see Table 4.9).

Next, to test that the sequence-to-sequence model is able to generalize to unseen questions using the synthetically generated data, we separate out 5 % from the synthetic training set for a temporary test set. We select the questions for this test set so that there is no exact overlap in templates between the training set and the test set. For example, we would include the questions generated from the templates 'Where is the tallest $\langle a \rangle$?' and 'What is the oldest $\langle a \rangle$ in $\langle b \rangle$?' into the training set and questions from 'What is the tallest $\langle a \rangle$ in $\langle b \rangle$?' into the test set.

Table 4.10 shows the BLEU score results for training and evaluating on the synthetic test data that we created. We can see that the Transformer model can generalize to new templates at test time better than the LSTM model.

Execution query results We compare the models in the hard transfer learning setting on the test part of the WebQSP-WD dataset (the same split as in Section 4.4.6). As introduced in Section 4.3, WebQSP-WD is a Wikidata compatible version of the WebQSP dataset, which was originally used with a different knowledge base. It is a heterogeneous knowledge base question answering dataset that has both simple questions that can be answered with a single relation and complex questions that require multiple relations and constraints. We remind, that the WebQSP-WD test set contains 1033 instances (Table 4.1).

We generate a SPARQL query for each question in the test set and evaluate it on

	P	R	F
Transformer (T)	0.0737	0.0719	0.0728
Transformer (T+G)	0.0762	0.0808	0.0784
ExpGGNN	0.3203	0.3752	0.3456
+Transformer (T)	0.3299	0.3751	0.3510
+Transformer (T+G)	**0.3386**	**0.3804**	**0.3582**

Table 4.11: Query execution results on the WebQSP-WD test set for the Transformer sequence-to-sequence models. The results are reported for the sequence-to-sequence models in isolation and with the ExpGGNN semantic parser as a back-off (see Figure 4.15). We use either the template generated questions (T) or the whole synthetic dataset including the questions generate by a neural model (G) for training.

Wikidata to retrieve the answer. The sequence-to-sequence models are trained on a mixed synthetic dataset and applied on the WebQSP-WD. We compare against the best semantic parsing model from the first half of this chapter (ExpGGNN). We only use the Transformer model in these experiments, since it has consistently outperformed the LSTM variant in the BLEU score evaluation.

We report results for the sequence-to-sequence models and their combination with the semantic parsing model. We use a straightforward combination procedure, where the output of the Transformer is used whenever the generated SPARQL query generates an answer and the output of ExpGGNN is used otherwise (e.g. when the SPARQL generated by the Transformer was invalid, see Figure 4.15).

The goal of our experiments is to test if a task-agnostic sequence-to-sequence architecture can be used to generate SPARQL queries and to approach knowledge base question answering. We use the previous semantic parsing model (ExpGGNN) as a point of comparison and do not re-train it on the synthetic generated training data. However, the automatic data augmentation could be a promising future direction to improve the performance of the semantic parsing models as well. We leave this direction for future work.

In Table 4.11, we show the Transformer results for query execution and the results for the combination of the sequence-to-sequence method with the ExpGGNN baselines. Transformer (T) and Transformer (T+G) are the weakest setups in the evaluation. These variants are an application of the sequence-to-sequence architecture without the semantic parsing back-off strategy. Transformer (T+G) generates a query that does not retrieve any answer in 79 % of the test cases. It hurts the overall recall of the sequence-to-sequence models and also the precision, as we take the precision to be 0 for an individual question if there is no answer produced. This results in a much lower F-score for the Transformer model compared to the semantic parsing model.

On the questions where Transformer (T+G) has generated a SPARQL query that produces any answer (the remaining 21 % of the test data), it achieves 0.3627 F-Score. It is important to note that the Transformer was trained on artificial data and does not rely on in-domain data the same way as the ExpGGNN does. We primarily focus on the application of the Transformer models on top of the semantic parser for the rest of the analysis.

We use the output of the sequence-to-sequence model whenever an answer is successfully retrieved and back off to the semantic parsing pipeline in other cases. We see in Table 4.11 that the combined system (ExpGGNN+Transformer) outperforms the best semantic parsing result for ExpGGNN (cf. Table 4.3 in Section 4.4.6). We also note that training the Transformer on the predominantly template generated data (T) already increases the F-score and the neural question generation (T+G) further improves the result.

A sequence-to-sequence model in combination with a semantic parser as an ensemble architecture does not only improve the overall question answering accuracy, but also leads to a faster response. The speed of response can be a critical consideration in the context of online question answering systems such as Ask Wikidata[14] or virtual assistants (such as Siri or Alexa) that can answer world knowledge questions. Since the Transformer is an end-to-end model and does not use the information from the knowledge base for entity linking or relation filtering, the combined architecture is also approximately 25 % faster than the pipeline alone at prediction time. The Transformer covers 21 % of the test data and has a negligible runtime per example compared to the semantic parsing pipeline. Whereas our semantic parsing approach requires around 2-3 seconds to answer a single question on the common hardware using a single GPU, the end-to-end Transformer model retrieves the answer in under 0.1 second. The semantic parser spends a disproportional amount of time for repeated queries to the knowledge base, which are avoided by the proposed Transformer-based architecture.

4.6.4 Error Analysis

We have manually analyzed the cases, when the Transformer model has not retrieved any answer (79 % of the test cases)[15]. A syntactically incorrect SPARQL query, which cannot be executed against Wikidata, causes 5 % of the cases without an answer. 60 % of the time the model did not retrieve any answer, because it has generated a query with too many constraints or a wrong relation. The rest of the errors are the contradictions between the query generated by the sequence-to-sequence model and the knowledge base schema, which also result in an empty retrieved set. In that cases our model generates a relation that is semantically very similar to the question, but is used differently in the Wikidata schema. For example, Wikidata encodes book authors with the AUTHOR relation and authors of musical compositions with COMPOSER. Being semantically similar, they are often

[14] `https://tools.wmflabs.org/bene/ask/`

[15] There are no test questions in WebQSP-WD, where an empty answer is the correct result.

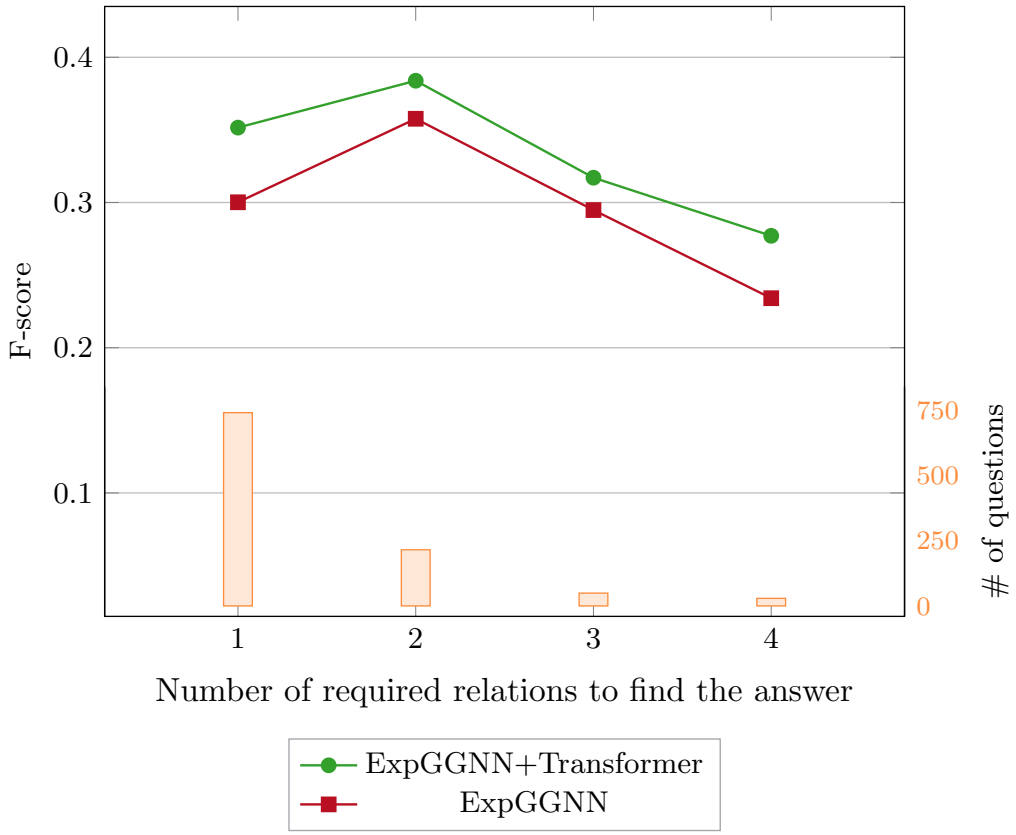

Figure 4.16: Evaluation results on the WebQSP-WD test set by the number of relations needed to find the correct answer for the Transformer model and ExpGGNN. The Transformer sequence-to-sequence model is used in combination with the ExpGGNN semantic parser as a back-off. The bars show the number of questions in the data in each category.

confused by the Transformer model, yet they are not interchangeable and are only compatible with the respective entity types.

The strategy not to return any answer if a model struggles to understand a question is to some extent justified to optimize for precision of the returned answers. It is reasonable to assume that no answer is better than a wrong answer in the context of the real-world question answering applications. The pipeline models (including the semantic parsing approaches), on the contrary, optimize for recall and select only from the relations that would result in a non-empty answer. That is, the pipeline communicates with the knowledge base to construct a query that would retrieve at least some entities. This slows down the pipeline approaches as they need to repeatedly send intermediate queries to the knowledge base.

In Section 4.4.6, we broke down the performance of our models by the number of relations that are needed to correctly answer the question. We break down the evaluation results for the combined Transformer+ExpGGNN model in the same way for further insight. Figure 4.16 shows the results by the number of the relations in the gold query for the ExpGGNN model and for the same architecture

Topic	F1	Correct Wikidata relation	Number of questions
spouse	0.1300	121	125
currency	0.3830	53	113
border	0.4070	26	40
capital	0.5050	35	35
zip code	0.0200	19	32

Table 4.12: Results for the sequence-to-sequence Transformer model for the most common question topics in the WebQSP-WD test set. The Transformer model generalizes well on the questions about a country capital, borders or currency. The spouse category suffers from incorrect entity linking although the relation type is predicted correctly 96 % of the time.

combined with the Transformer model. Our ExpGGNN semantic parser focuses predominantly on complex graph parsing and experiences a drop in performance on the simple one-relation questions. We see that the Transformer model improves the performance the most on the simple questions with one relation (the left most column in Figure 4.16), which is the largest category in the test set. Moreover, the combination with the Transformer architecture not only improves the overall performance of the knowledge base semantic parsing model, but also evens it out across the question categories.

We have manually looked into the Transformer model predictions to get an idea of the individual errors. Our analysis is organized into the most common question topics in the WebQSP-WD, which helps us to see if the model can handle different phrasings of the same question topic. We group test question into topics using a simple keyword search. We note the model performance on each topic and how often the corresponding Wikidata relation type was correctly identified in the generated query in Table 4.12.

In the 'capital' topic, the mistakes are mainly due to using the wrong entity or predicting one relation instead of two. For example, for "What province is Canada's capital located in?" the capital itself is returned instead of its location. On the questions about borders and neighboring countries the Transformer model is also able to include a correct query constraint (expressed with the LIMIT statement in SPARQL) if a number is given in the question (e.g. "What 5 countries border Switzerland?"). The most common source of errors for the 'spouse' category is the mismatch between the entity names in the WebQSP-WD dataset and in Wikidata. For example, Wikidata contains an entity 'Ronald Weasley' for a character in the Harry Potter book series. The same person is references as Ron Weasley in the dataset, which is hard for the sequence-to-sequence model to link correctly (in the semantic parsing pipeline the separate entity linking model is better equipped to handle such cases). Despite the low F-score, the Transformer model predicts the correct relation 96 % of the time for the 'spouse' category.

The Transformer model is also capable of answering questions about currency in around 40 % of the cases. Notably, this category of questions does not come up in the training data, but the word 'currency' is featured in other contexts, which presumably allows the model to generalize correctly.

We have demonstrated that the state-of-the-art Transformer model can be used to build a sequence-to-sequence knowledge base question answering architecture. To enable the training of the Transformer that requires a lot of training data, we have composed a large training dataset of synthetic questions. The resulting Transformer model is able to answer only 21 % of the test questions but achieves good accuracy on certain question categories. In combination with the semantic parsing pipeline, it improves the overall result and speed.

The persisting problem with the sequence-to-sequence approaches for mapping questions to SPARQL appears to be the limitation on the model's recall, since these models fail to retrieve an answer for every question in the dataset. One of the promising future directions is introducing template structure into SPARQL generation to prevent the model from generating syntactically incorrect queries and adding explicit type checks to prevent some of the schema incompatibility errors, akin to the recent advancements in text-to-SQL (Dong and Lapata, 2018).

4.7 Chapter Summary

In this chapter, we have used GGNNs to encode the structure of the target semantic representation for knowledge base question answering. We have shown that disregarding the semantic structure leads to a falling performance on questions that require complex semantic representations to get the correct answers. Our GGNN-based architecture successfully models the structure of semantic graphs. We advance the system further by adding Levi graph transformation and size normalization for the final ExpGGNN model.

We have compared the performance of graph-based models against the previous work and against our own baseline models on the WebQSP-WD benchmark and have broken down the results by question complexity. The analysis has shown that the suggested graph architectures do not have the same drop in performance on complex questions that can be observed in previous work and produce better overall results. Peng et al. (2017) and Yu et al. (2017) have attempted to incorporate entity linking into a feature-based question answering model. In the future, it is reasonable to integrate entity linking with ExpGGNN, as well.

After the successful evaluation of the ExpGGNN semantic parsing architecture, we have followed up with two directions for semantic parsing: applying ExpGGNNs on different related tasks and exploring an alternative architecture in a form of sequence-to-sequence models.

We have explored document question answering as a new application for knowledge base semantic parsing. We have successfully combined the structured

grounded semantic representations produced by ExpGGNN with a state-of-the-art text comprehension approach and have improved the performance on the TriviaQA open-domain document question answering dataset. Our semantic parsing component uses the ExpGGNN architecture to build a knowledge base semantic graph. The semantic graph encodes constraints in the input question, which are often missed by the current text comprehension models. We use the produced semantic graph to filter out the output of the document question answering model. In that manner, the suggested approach dynamically integrates external knowledge and is easy to interpret by humans.

With the reported experiments, we provide an empirical evidence that structural information from a knowledge base can mitigate the mistakes made by a common text comprehension model, as also adds an additional layer of interpretability to such models. The two main components of our TriviaQA system are trained independently and we rely on the optimized confidence threshold to mix the outputs. In the future, it is important to develop systems with a learned policy of selecting either the text comprehension system output or the output of the semantic parser.

To explore an alternative to a semantic parser, we have applied the sequence-to-sequence Transformer architecture on the WebQSP-WD dataset. We have demonstrated for the first time that the state-of-the-art Transformer architecture can be used to build an end-to-end knowledge base question answering system. We have composed a large dataset of synthetic questions with sufficient coverage for training. The Transformer model individually performs worse than our ExpGGNN semantic parser, but we show that it is possible to use the Transformer in combination with ExpGGNN to improve the performance on simple questions with one semantic relation. Our Transformer question answering architecture is also much faster than the semantic parsing pipeline.

Let us return to the two research question we have selected for this chapter:

RQ2 How to encode structured semantic representations for effective grounded semantic parsing?

RQ3 Can the knowledge base linking methods and the grounded semantic parser generalize to new data and be embedded into natural language understanding applications?

We have developed and empirically tested effective methods for encoding the structure of the semantic knowledge representations with GGNNs (RQ2). These methods have improved the empirical performance on the knowledge base question answering task and the text comprehension task (RQ3). Furthermore, to pave the way for future work, we have explored a sequence-to-sequence alternative to producing structured knowledge representations in a form of knowledge base queries. The sequence-to-sequence methods demonstrate promising properties (they are fast and potentially task agnostic), but do not yet approach the performance results of our GNN semantic parser, the ExpGGNN model.

Chapter 5

Interactive Instance-based Analysis

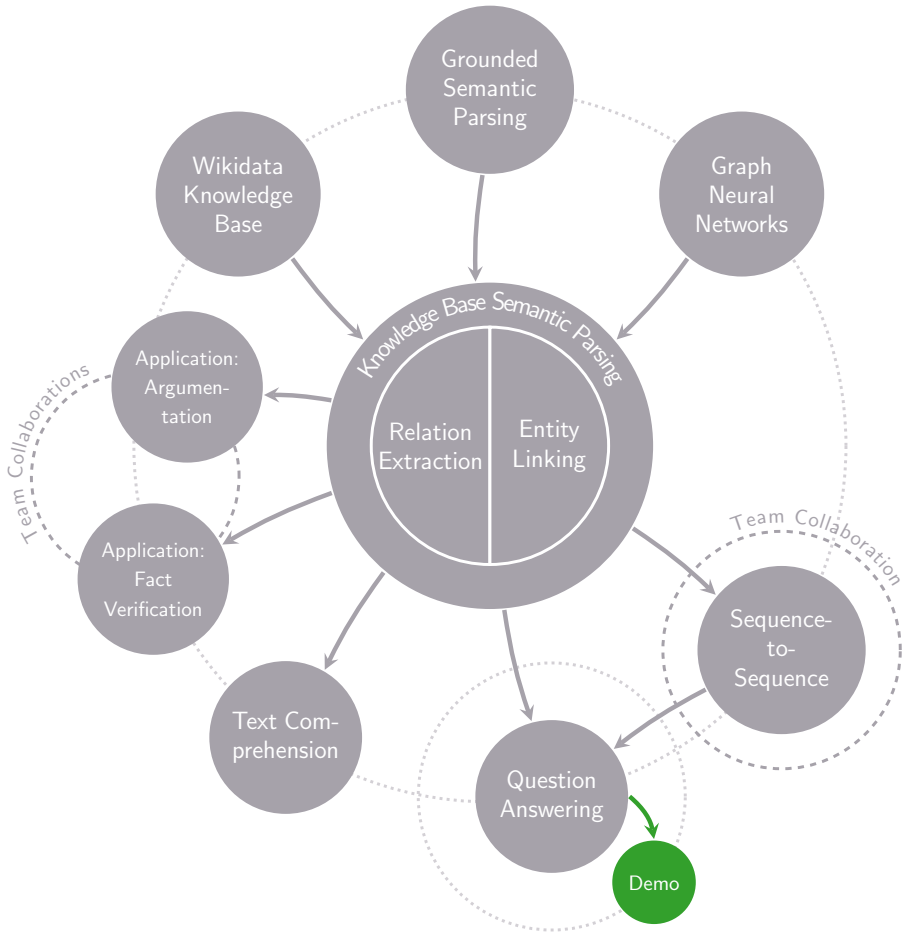

Figure 5.1: Chapter 5 in the thesis diagram. This chapter describes an interactive evaluation analysis tool for the semantic parsing approach to question answering (the green circle at the bottom).

Most approaches to knowledge base question answering are based on semantic parsing. In the previous chapters, we have introduced new methods for linking texts to a knowledge base and a knowledge base semantic parser for the question answering task. In this chapter, we present a tool that aids in the analysis and debugging of a complete question answering pipeline that constructs a structured semantic representation for the input question.

The existing work on the analysis of knowledge base question answering systems has focused on building knowledge base interfaces or evaluation frameworks that unify multiple datasets (Miller et al., 2017). However, none of these frameworks allows a researcher to conduct an instance-based error analysis and interactively track how each example is being processed by the knowledge base question answering system.

The primary objective of our tool is to enable interactive debugging of pipeline predictions on individual examples (questions) and to simplify manual qualitative error analysis. Additionally, our tool showcases a full question answering pipeline that combines the input pre-processing, entity-linking, semantic parsing and answer retrieval steps. We put together the components developed throughout this thesis to make the final prototype for the tool demonstration. The tool combines an interactive interface for the question answering pipeline and visualization views for the outputs at the different stages of the pipeline. The user interface structure follows the conceptual steps of the grounded semantic parsing to make it intuitive for researchers (we refer to Figure 2.11 and to the discussion in Section 4.1 for the prototypical semantic parsing pipeline).

Our interactive interface helps researchers to understand the shortcomings of a particular model, to qualitatively analyze the complete pipeline and to compare different models. We use a set of sit-by testing sessions with users to validate that our interface design is intuitive and easy to use (Rubin and Chisnell, 2008). We also report on the practical issues of constructing the full-stack question answering systems.

To summarize, we present a modular debugging application for knowledge base question answering that can be used to manually evaluate the main steps of a semantic parsing pipeline. Our system focuses on the analysis of individual examples and offers a detailed view of possible causes of errors, so that individual error propagation cases can be identified. A demo instance of the developed tool with the default question answering model is running at the following url: `http://semanticparsing.ukp.informatik.tu-darmstadt.de:5000/question-answering/`. Our code is freely available here: `https://github.com/UKPLab/emnlp2018-question-answering-interface`.

5.1 Motivation

The task of knowledge base question answering is to find a set of entities in a knowledge base to answer a natural language question. For example, for a question "Who played Princess Leia?" the answer, 'Carrie Fisher', could be retrieved from a general-purpose knowledge base, such as Wikidata which we have used in our experiments in previous chapters. A successful knowledge base question answering system ultimately provides a universally accessible natural language interface to factual knowledge (Liang, 2016). In this chapter, we build a web-based tool with an interactive interface for the manual error analysis of the knowledge

base semantic parsing components on the question answering task. We have used this interactive tool to analyze the entity linker and the semantic parser developed in this thesis (see the error analysis in Section 3.2.7 and Section 4.4.7).

Knowledge base question answering requires a precise modeling of the question semantics through the entities and relations available in the knowledge base to retrieve the correct answer. We have discussed in Chapter 4 that it is common to break down the question answering task into three main steps: entity linking, semantic parsing or relation disambiguation and answer retrieval. This three-step approach has been exhibited by the most recent works (Berant and Liang, 2014; Reddy et al., 2016; Yih et al., 2015; Peng et al., 2017; Luo et al., 2018) and in our own approach. We show in Figure 5.2 how the outlined steps lead to an answer on an example question and how the output of each step is reused in the next one. We take this workflow as a basis for the layout of the user interface of our tool.

A multi-step semantic parsing pipeline poses particular challenges for error analysis, since unique errors can arise at different processing stages and lead to error propagation. In Chapter 4, we have built a single ExpGGNN model to cover both the relations extraction and semantic representation construction in a single step, but entity linking remains a separate step in the pipeline. With our online tool, we aim at supporting the evaluation of question answering systems and helping to identify problems on a fine-grained level by analyzing each individual question and how it is handled by every step in the pipeline.

Some frameworks have been recently introduced to streamline the evaluation of knowledge base question answering systems. The ParlAI framework focuses on building a unified interface for multiple question answering datasets (Miller et al., 2017), while GerbilQA[1] introduces evaluation of individual steps of a semantic parsing pipeline. However, none of them addresses an interactive debugging scenario that can be used by the researchers to do instance-based error analysis. This is especially relevant in the context of such benchmarks as QALD (Unger et al., 2016), where each individual question is meant to test a particular aspect of the system and debugging each example is crucial for understanding of the system performance.

Other tools have focused on building an infrastructure to support the development of online systems for knowledge base question answering. The Ask Wikidata tool[2] offers an easy way to post queries to Wikidata via a web-based interface, though the tool does not include a semantic parsing model and relies on the user to manually disambiguate a question. The WDAqua project[3] has produced a speech-based plug-in interface (Kumar et al., 2017) and the Qanary specification for question answering systems (Singh et al., 2016). These tools follow the described prototypical steps of the question answering pipeline, but do not facilitate the interactive instance-based evaluation that is the main aspect of this work.

[1] http://aksw.org/Projects/GERBIL.html
[2] https://tools.wmflabs.org/bene/ask/
[3] http://wdaqua.eu

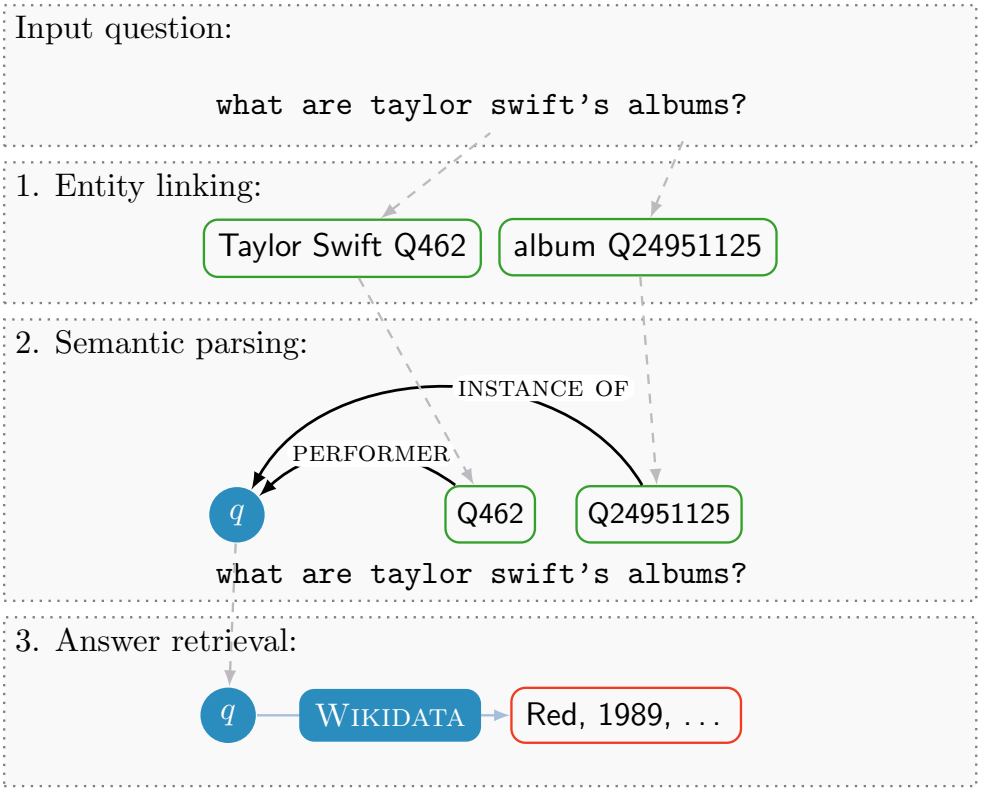

Figure 5.2: Typical steps undertaken by a semantic parsing system for the question answering task. Gray dashed arrows show how the output of the previous step is passed into the next one.

5.2 Requirements

The first step of every question answering approach is entity linking, which identifies entity mentions in a question and links them to entities in the knowledge base. In Figure 5.2, two entity mentions are detected in the input question and are linked to the knowledge base entities. We have discussed in Section 3.2 that entity linking is a common source of errors in a question answering system and it is therefore important to be able to inspect the output of this step in the pipeline.

The entity linking step is followed by the semantic parsing that consists of combining the extracted entities into a single structured meaning representation. The entities are connected with semantic relations to a special question variable that denotes the answer to the question or to intermediate variables (the approach adopted in Yih et al., 2015; Luo et al., 2018 and in our implementation).

Given the ambiguity of the natural language, a semantic parsing model constructs multiple representations that can match the question and assigns probabilities to them (Liang, 2016). As we have shown in Section 4.4, it is common to learn a vector encoding for the question and the structured representations and then use a similarity function to compute the probabilities. The most probable structured

representation is then translated into a query and used to extract the answer from the knowledge base. The overall process is called grounded semantic parsing (see Section 2.2).

Some approaches circumvent building a structured representation and instead directly compose vector encodings of the potential answers (Dong et al., 2015). This is a less common architecture type for knowledge base question answering and as we have discussed in Section 4.1, it is not as performant as grounded semantic parsing. Therefore, we focus on semantic parsing approaches while developing our interface.

The described pipeline lets us outline the main requirements for an interactive debugging tool:

1. It needs to represent all stages of the question answering pipeline in a sequential manner to let the user identify where the error occurs and how it propagates.
2. It needs to account for the specific properties of semantic parsing approaches to knowledge base question answering, such as structured semantic representations.
3. It needs to include a visual component that enables the user to inspect if the model has learned meaningful vector representations for individual questions.

5.3 System Overview

Our system consists of a web-based front-end and a set of back-end services that communicate through HTTP REST API (see Figure 5.3). The front-end contains the interactive debugging user interface (described below in Section 5.4). A separate request is sent to the back-end service for each processing step of the question answering pipeline. With this setup, the user is able to see the results as they are being delivered by the back-end services. The front-end is responsible for aggregating and visualizing the information after each step. In case any service fails, a partial result from the previous steps would be available to the user. The interplay of the iterative requests to the back-end and the visualization of the intermediate responses in the front-end make clear when the question answering pipeline fails and at which stage. It also enables to show the user partial processing results if the full processing fails and indicate, which modules have completed their part successfully.

The back-end services include the pre-processing, the entity linking and the semantic parsing modules. The pre-processing module performs tokenization and part-of speech tagging using the Stanford CoreNLP toolkit (Manning et al., 2014). The entity linking module recognizes mentions of the entities in the input question and links them to their knowledge base referent. We include a default model for

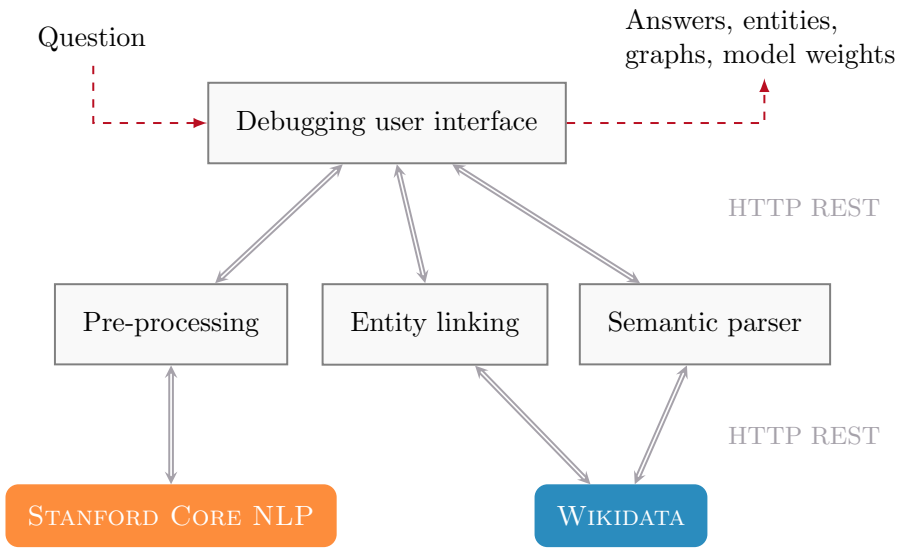

Figure 5.3: Overview of the interactive evaluation tool architecture. The main components of the tool: Pre-processing, Entity linking, Semantic Parser and Debugging user interface. The main components interact with the external services: Stanford Core NLP server and the Wikidata SPARQL endpoint.

```
entity.linking:
  model: models/elmodel.pkl
  max.entity.options: 5
  max.ngram.len: 4

model:
  model.file: models/qamodel.pkl

evaluation:
  max.num.entities: 3
  beam.size: 10
  ...

wikidata:
  backend: localhost:8890/sparql
  timeout: 10
  ...
```

Listing 5.1: A snippet of the YAML configuration file for the interactive debugging system with the main modules and settings.

entity linking on Wikidata, a freely available implementation from Section 3.2. It is possible to replace the entity linker with any alternative service that implements the same HTTP REST API.

The semantic parsing module includes the construction of structured semantic representations and a learned model that selects the correct representation. The integrated model uses a convolutional neural network architecture to learn vector encodings for questions and GNNs for semantic representations (see Section 4.4).

We provide an integrated model for demonstration purposes, while the main purpose of the tool is to enable manual evaluation and comparison of further models. We define an HTTP REST API, which the user needs to implement to integrate their own model into the tool and include the description with the published code.

Finally, using the knowledge base query provided by the semantic parsing module, the front-end retrieves the answers from the knowledge base. The back-end modules can be configured using a YAML properties file (see Listing 5.1 for an example configuration). The properties file directly exposes the parameters of the default models for entity linking and semantic parsing that are included with the demo version. It also allows to change the evaluation and inference parameters as well as the knowledge base SPARQL back-end.

We implement the user interface and the back-end services with modern web technologies. The front-end uses JQuery[4], D3.js[5] and Bootstrap[6] open-source libraries to compose the dynamic interface. The back-end services are implemented in Python with Flask[7], a lightweight web application framework. It is configurable and can be further easily extended with other models for entity linking and question answering. A new question answering semantic parsing model can be integrated either as a Python module or as a separate REST service that adheres to the HTTP REST API defined in the code. To communicate the results to the front-end, the service has to send the response in the defined JSON format. We refer to the published code repository for additional details on the implementation.

5.4 User Interface

The interactive web-based user interface is the central element of our tool. We have designed the interface for an expert user and have considered the following user traits (Raskin, 2000): a background in knowledge base question answering, a knowledge of programming languages, an interest in manual error analysis of a semantic parsing question answering pipeline.

The user interface is modeled after the prototypical question answering pipeline as described in Section 5.2. Each step of the pipeline is represented as a separate block in the interface (see the complete user interface depicted in Figure 5.4). We call a *block* a visually separated part of the interface that corresponds to a single task in the semantic parsing pipeline or exposes a single analysis method (such as the 'Entity linking' block or a 'Vector representations' block in Figure 5.4). That is, the represented model of the user interface directly corresponds to the implementation model of a prototypical semantic parsing question answering

[4] https://jquery.com
[5] https://d3js.org
[6] https://getbootstrap.com
[7] https://palletsprojects.com/p/flask/

Figure 5.4: The complete unrolled user interface of the interactive evaluation tool for knowledge base semantic parsing analysis. We use an example question "Who played Luke Skywalker in Star Wars?".

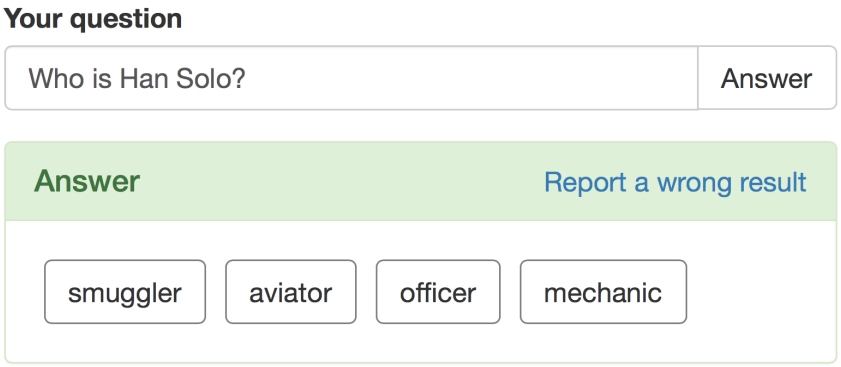

Figure 5.5: The main input field and the answer block of the interactive evaluation tool.

pipeline. Such design choice is appropriate for tools aimed at domain experts who already have a clear mental model of the underlying processes and it makes the user interface understandable for the first time users (Cooper et al., 2007).

Consequently, the interface is divided into blocks that correspond to the steps of the question answering pipeline. There are base blocks that are fixed and cannot be removed from the interface (e.g. the input question block) while other blocks can be hidden and depend on the output of the corresponding tools. We explain the individual blocks below.

Input question and answer block The first block consists of the input question field and the answer area (Figure 5.5). When the user first loads the interface, only the input question field is presented. This avoids the confusion as to what is the starting point of the interactions with the system. Further elements appear only when the corresponding results are returned. For example, the answers area is only shown when the complete processing is finished. We also include the information on the overall processing time of the user request.

Although the answer retrieval from the knowledge base is the last step of the pipeline, we put the answer area right below the input question field. This design choice makes it easy to see right away if the pipeline has processed the question correctly.

Entity linking block In this block, we list all identified entity mentions in the input question and the top five entity disambiguation candidates (the information is visually grouped under 'Recognized entities' and 'Entity linkings', see Figure 5.4). The entity candidate with the highest score is automatically selected and forwarded to the semantic parsing model at the next step.

The list of entity disambiguation candidates is interactive and the user can select all or none candidates for each entity mention. In case multiple candidates are selected, all of them are sent to the semantic parsing system as separate entities.

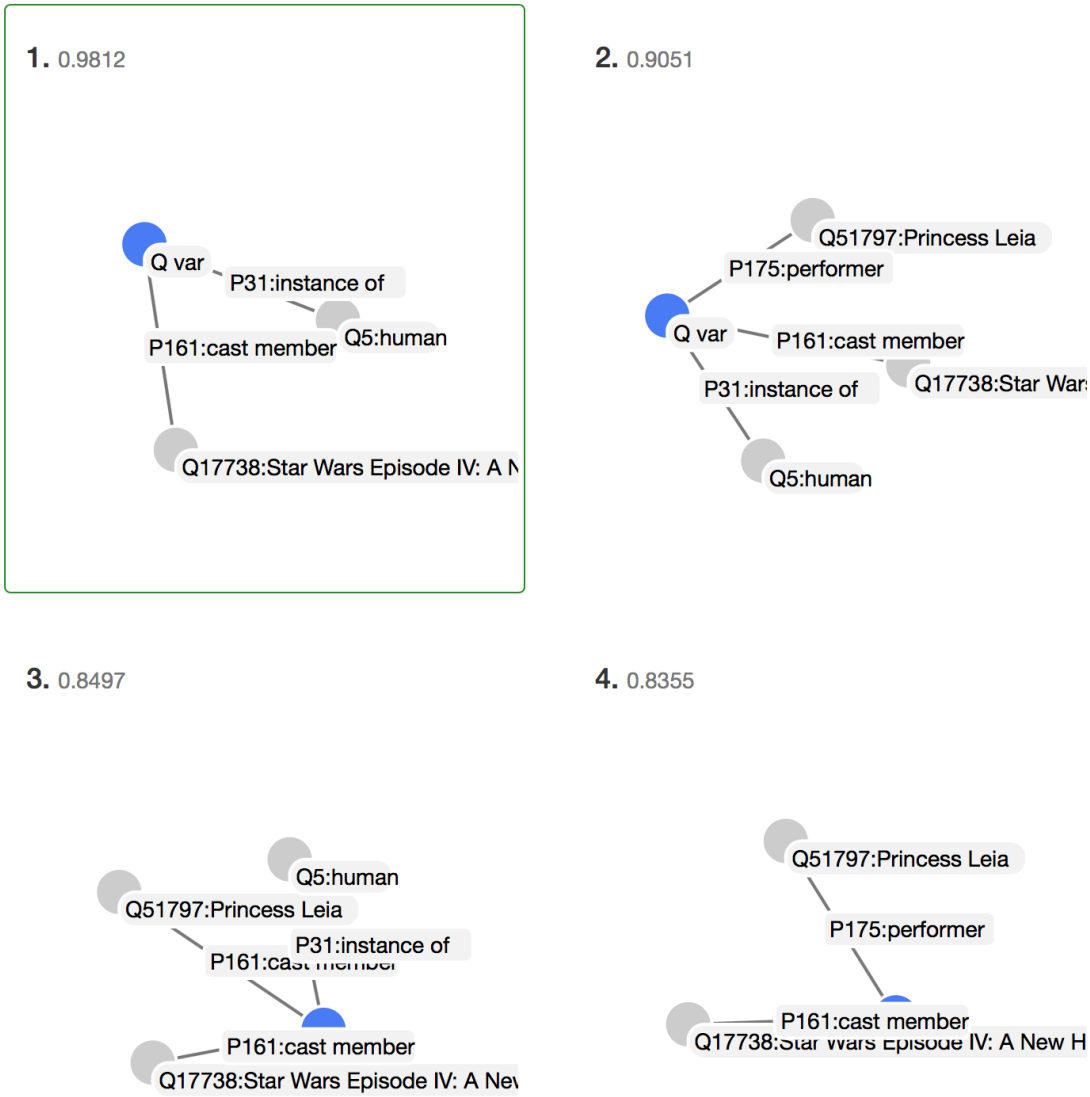

Figure 5.6: The semantic graphs block of the interactive analysis tool for the question "Who played Princess Leia in Star Wars?".

This lets the user correct potential entity linker mistakes by selecting some other than the top disambiguation candidate and continue to debug the rest of the pipeline.

Semantic graphs A semantic parsing model like the one we describe in Section 4.4 constructs a semantic graph of the question. It is often possible to retrieve a set of the top hypotheses generated by the model and inspect them. This provides an insight into the model performance, in particular if the top choice of the model was not the correct semantic representation. In our implementation of the semantic parsing for question answering, we generate possible semantic representations for the input question and rank them with the neural model. For the purpose of the tool demonstration, we store the ranked list of the generated

representations.

We visualize the structured semantic representations that the question answering model generates as graphs (as shown in Figure 5.6). That is the most common way to visually depict structured semantic representations, as is evident in previous work (Yih et al., 2015; Reddy et al., 2016; Luo et al., 2018) and in this thesis. A semantic graph consists of a question variable node that denotes the answer to the question, knowledge base entities and knowledge base relation types (see the more detailed description in Section 4.2).

We use circles to depict entities and solid lines for relations. In each graph the question variable node is represented with a high-contrast blue circle. Since most relations are attached to the question node, this makes easier to parse the structure of the graph on the first sight. Additionally, since each graph has only one high contrast node, the user can identify quickly how many graphs have been composed for the input question. For example, Figure 5.6 shows four semantic graphs for the question "Who played Princess Leia in Star Wars?". In this instance, the model selects an incorrect graph as the top one (highlighted with a green frame) and retrieves all cast members of the Star Wars movie. The correct graph would be the second one, that also includes the entity 'Princess Leia'. Using the interactive tool, we can quickly notice that although the model has produced a wrong answer, the correct representation was a second top choice and the similarity score for it is only slightly different from the top one (0.98 vs. 0.91).

Representation analysis The visual inspection of the learned vector representation in the two final blocks makes it possible to identify possible implementation or training errors in the question answering model. Once an error is attributed to the learned model, the user can continue to analyze the model in the tool that is the most appropriate to inspect a particular statistical model architecture (e.g. TensorBoard[8]).

The visual inspection of high-dimensional representations is a challenging problem. We offer a starting point for the model weight analysis on individual examples and encourage the user to continue to dive deeper with the other dedicated tools. We offer two orthogonal ways to inspect the learned weights in the model: at the token level of the input question and at the level of the vector encodings for the input question and for the knowledge base semantic relations ('Token encodings' and 'Vector representations' in the lower part of Figure 5.4).

The token-level representations block visualizes the weights computed by the model for each input token of the question. This kind of visual analysis is helpful to identify if the model is learning meaningful token representations. We visualize each token vector as a row of vertical lines with varying visual attributes. We rely on the shade and saturation visual variables to encode the computed vectors. Each vertical line corresponds to a vector dimension and the darker saturated

[8] https://www.tensorflow.org/programmers_guide/summaries_and_tensorboard

colors denote a higher numerical value. In Figure 5.4, one can see that the model is assigning the highest weights to the main entity in the question (the '$\langle e \rangle$' token on the left of the visualization).

The second representation analysis block places the vector representation of the question and of the semantic relations on a 2D-plane using the t-SNE vector visualization method (van der Maaten and Hinton, 2008). We show all knowledge base semantic relations that were available for the semantic graph at the construction stage. The vectors of the relations that appear in the top generated semantic graphs are rendered as red circles. The user can zoom in and out to compare the position of the question vector to the surrounding relations.

5.5 User Study

The user interface for our interactive analysis tool was designed with the requirements that were outlined in Section 5.2. As domain experts, we were personally interested in applying the developed tool to an analysis of a semantic parsing system for question answering. To verify that the interface and the designed interactions are in line with expectations of the first-time users, we have conducted a set of brief sit-by testing sessions. Sit-by sessions are usually used for exploratory situations and gathering first impressions about the design of a product (Rubin and Chisnell, 2008).

In the user study, we aimed to answer the following questions: can a person familiar with the knowledge base question answering and semantic parsing use the developed tool independently? Does the tool make it possible to manually identify errors in a question answering pipeline? Two participants with background in natural language processing and linguistics were asked to perform a simple analysis task while a moderator was sitting near them and monitoring the progress. The participants were asked to input a list questions of their choice into the tool and tell if the model succeeded in answering them. We have asked the participants to find at least 3 questions that were incorrectly processed by the semantic parsing pipeline.

For the incorrectly processed questions, we have expected participants to identify the stage of the pipeline that caused the error. During the sit-by sessions, we were able to confirm that the user interface is intuitive and easy to use. Both participants were able to complete the task in under 10 minutes and could point out at what stage an error has occurred for all input questions. The representation analysis instruments, on the contrary, have proven to be the least intuitive element of the interface. Although the participants could attribute the error to the model, they were unable to say if the learned vector representations were meaningful based on the provided visualization.

5.6 Chapter Summary

In this chapter, we have presented an interactive analysis tool for semantic parsing approaches to knowledge base question answering. This tool is focused on the analysis and debugging of individual examples to facilitate the qualitative error analysis. We have incorporated a standard pre-processing toolkit into the tool and made possible to integrate multiple entity linking and semantic parsing models. For the demonstration of the tool, we have used the components that were developed through this thesis: the entity linker described in Section 3.2 and the semantic parser from Section 4.4. We have used the demo version of the tool for the error analysis of the respective components while conducting the experiments (see Section 3.2.7, Section 4.4.7 and Section 4.5.5). To that end, the interactive evaluation tool has contributed to all research question that we have considered in this thesis.

We have employed the interface design guidelines from Raskin (2000) to make sure we create an intuitive instrument for expert NLP users. We have started by defining the main requirements for an instance-based evaluation tool and then demonstrated how the different aspects of the designed interface fulfill them. Our tool enables researchers to explore and qualitatively analyze a developed question answering pipeline. The included default models for entity linking and question answering make it possible to replace only one of the components with a new module or further modify the evaluation setup.

We used sit-by sessions to verify the design choices and to assess the usability of the tool. The results indicate that our design is intuitive and easy to use for qualitative analysis. We note that the vector encoding analysis components do not offer a complete picture for the user and have a more limited usability. The visual analysis of vector encodings is a challenging problem and we provided a first insight for the user into possible encoding learning issues.

Chapter 6

Conclusion

This chapter is meant to summarize the work that we have presented in this dissertation and to look ahead towards the next challenges and trends. First, we review our contributions and return to the answers to the main research questions that we have posed in Chapter 1. Second, we look beyond the topics that we have touched in this manuscript and suggest how our work can be extended and continued in all of the main directions.

6.1 Summary

The automatic processing of textual data and the explicit representation of the meaning of the natural language remain challenging. When humans exchange information in the natural language form, we rely on the commonly shared world knowledge to interpret the utterances of another person. An successful NLP system should be able to interpret the meaning of the human language in a broader context of the already available world knowledge. In this thesis, we have investigated how an open-domain knowledge base can play the role of the interpretation context for an NLP algorithm and how the process of linking a natural language text to an external knowledge base can be improved.

The process of representing the meaning of language in a structured form and linking it to an external resource is called grounded semantic parsing. We have chosen to build upon logic-based formalized meaning representations, since they offer a framework for representing a text with an expressive semantic formalism that can be grounded in an external knowledge base and supports many extensions for inference, processing modalities and more.

We have used the Wikidata knowledge base as the main resource for world knowledge and for grounding semantic representations. Throughout this work, we have built methods for linking texts to Wikidata and for constructing grounded semantic representations and applied them to downstream NLP applications, such as question answering. We follow the elements of the overview diagram in Figure 6.1 as we go along with the retrospective of this thesis.

To start with, we have provided theoretical and methodological background on grounded semantic parsing with Wikidata in Chapter 2. We have introduced the Wikidata knowledge base and its main properties, such as direct links to Wikipedia. We have defined a grounded semantic representation that uses first-

Figure 6.1: A schematic representation of the topics discussed in the thesis and their connections as a thesis diagram (repeated from Figure 1.3). Upper orange circles (Chapter 2): theoretical and methodological background on grounded semantic parsing with Wikidata. Blue semicircles at the center (Chapter 3): linking the relations and entities in text to Wikidata with rich context encoding. The large red circle at the center and the red circles in the bottom (Chapter 4): combining the linking methods into a semantic parser and applying it to question answering. The green circle at the very bottom (Chapter 5): an interactive evaluation analysis tool and a demo for the semantic parsing approach to question answering.

order logic syntax and can be interpreted in Wikidata. These semantic representations are constructed by creating a graphical structure that encodes the meaning of a sentence and by linking its elements to the knowledge base relations and entities. Thereby, we have explained that linking texts to a knowledge base is the main component of constructing a complete grounded semantic representation. We have also sketched an approach that links relations in a text and constructs the structure of the representation in a single step. In the final section of Chapter 2, we have introduced a powerful neural network type of GGNNs, which allows us to directly process graphical structures of semantic representations.

In Chapter 3, we have discussed new methods for linking parts of a text to knowl-

edge base elements. Methodologically, we focused on the efficient usage of the sentential and knowledge base context for relation extraction and entity linking. For relation extraction and disambiguation, we processed the information about relations in the same context and used it to make joint predictions of several relations per sentence. This goes against the wide-spread assumption of only one knowledge base relation per input sentence. We have improved the error rate by 24 % compared to a baseline that did not make use of other relations in the same sentence. For entity mention detection and linking, we have noted the importance of the different levels of context and proposed to aggregate together the character, word, knowledge base and relational information. Our new entity linking architecture resulted in an average 8 % improvement of the final F-score on short-text data compared to a state-of-the-art entity linker. Subsequently, we have extrinsically evaluated the entity linker on two NLP applications: argument reasoning comprehension and fact verification. We have observed in the both scenarios that enriching the input with knowledge base entities aided in text understanding. For argument reasoning comprehension, the entity knowledge is helping to draw the decision between two reasoning chains, when there is one or two entities in the input. For fact verification, entity linking forms a crucial first step in the pipeline, which is responsible for finding relevant documents to verify the facts in the input.

The relation extraction and entity linking form the foundation of our grounded semantic parser in Chapter 4. We have used the entity linker to find the knowledge base entities in the input that would be used to construct a graphical semantic representation. We have taken into account the outcome of the relation extraction experiment in Chapter 3 and built a system that jointly disambiguated all relations in the input. Our approach links all relations and constructs the semantic representations in one step, as we have sketched in Chapter 2. We have used a single GGNN architecture to process all elements of a semantic representation. The original GGNNs are not able to incorporate knowledge base relations by default. We have extended their formulation to include knowledge base relations and used the updated GGNNs to select the correct semantic representation for the input text. In the previous work, Yih et al. (2015) and Luo et al. (2018) predicted relations for grounded semantic parsing separately and focused on choosing only one main relation. By evaluating a semantic representation as a whole with GGNNs, our approach is able to accurately compose semantic representations out of several relations and entities.

Answering questions about facts is the main NLP application that we have considered in this thesis. We have applied our grounded semantic parser on this task. We have shown on a knowledge base question answering dataset and on a text comprehension dataset for English that our GGNN-based architecture can improve upon the state-of-the-art results. The improvement is most prominent on the questions that should be encoded with multiple relations. In the final section of Chapter 4, we have contrasted our GGNN architecture with a sequence-to-sequence model that directly generates queries to the knowledge base. We have

shown that the direct-to-query approach might be faster, but has lower accuracy than our GGNN semantic parser.

In Chapter 5, we have described an interactive evaluation system for knowledge base semantic parsers. We have employed the developed interactive tool to aid with the manual analysis of our entity linking and GGNNs semantic parsing approaches that we have reported on in Chapters 3 and 4.

In the present dissertation, we have focused on three research questions regarding the grounded semantic parsing and question answering with Wikidata. We now recall and collect our answers to these questions.

RQ1 What are the challenges for linking a text to an external knowledge base and how can the linking methods be improved?

To answer the first part of **RQ1**, we have started by analyzing the existing grounded semantic parsing pipelines in Reddy et al. (2016) and Yih et al. (2015). We have described the major semantic parsing subtasks: syntactic analysis, entity and relation identification, ungrounded graph construction, linking entities and relations to the knowledge base. These steps are visually summarized in Figure 2.11. The disambiguation of entities and relations are the main linking steps from ungrounded to grounded semantic parsing. Hence, we have focused on each of those tasks before moving on to build a grounded semantic parser.

From the previous work, we have concluded that existing methods for relation extraction and entity linking lack a comprehensive modeling of the surrounding context. To answer the second part of **RQ1**, we have proposed a comprehensive modelling of the surrounding context, which improved both relation extraction and entity linking. We have introduced a **context-aware relation extraction** model that builds an aggregated weighted representation of all relations in the same context and makes joint predictions. We have created a new English Wikidata relation extraction dataset to test the proposed method against non-context-aware models. For entity linking, we have examined the challenges of applying it on question answering data, which is a common benchmark for grounded semantic parsing. We have determined that the current entity linking systems fail to perform consistently over different entity categories which are present in the question answering data. Consequently, we have argued that it is important to break down the entity linker performance by fine-grained entity categories. We have presented the **Variable Context Granularity network** for entity linking, which follows our context-aware paradigm and aggregates information from the sentential context and from the knowledge base.

We have answered **RQ1** by providing new methods to extract and disambiguate semantic relations and to link entities to Wikidata. Our methods explicitly model the surrounding context and improve upon the state-of-the-art approaches on the respective benchmarks.

RQ2 How to encode structured semantic representations for effective grounded semantic parsing?

To answer **RQ2**, we have presented an end-to-end method for grounded semantic parsing. We chose question answering as the main evaluation task. It is a common benchmark for grounded semantic parsers (Berant et al., 2013; Cheng et al., 2017; Reddy et al., 2017). Knowledge base question answering requires to link an input question to the knowledge base and to retrieve the answer. Therefore, the question answering task exactly matches the main goal of the present dissertation: to link the texts to external knowledge and thereby construct a grounded semantic representation that can be used for NLP applications.

Our semantic parsing approach to knowledge base question answering builds upon our findings from relation extraction and entity linking and combines them to solve the task of building a grounded semantic representation. From the relation extraction experiments, we have learned that it is crucial to extract the relations in the same context jointly. Therefore, we have unified the relation disambiguation and graph construction steps by defining a candidate semantic graph generation procedure and a graph evaluation function. The evaluation function encodes the candidate semantic graph as a whole and decides if the candidate matches the input question. The key to an effective evaluation function is a vector encoding that accurately reflects the different components of the structured semantic representation. We have adapted the GGNNs (Li et al., 2016) to encode structured semantic representations and have empirically shown that our method outperforms the previous semantic parsers, which do not encode the relations and the structure of a semantic graph together. **The graph network based semantic parsing model (ExpGGNN)** that we presented is the state-of-the-art method for grounded semantic parsing and knowledge base question answering with explicit semantic representations.

We have answered **RQ2** by providing a new GGNN-based method that encodes all information in a structured knowledge representation (entities and relation labels, relation directions and the structure of the graph) in a single model (ExpGGNN). We have shown the effectiveness of our approach by comparison against semantic parsing baselines that do not use GNNs on the knowledge base question answering task.

RQ3 Can the knowledge base linking methods and the grounded semantic parser generalize to new data and be embedded into natural language understanding applications?

To answer **RQ3**, we have applied the developed method for linking text elements to Wikidata and for grounded semantic parsing to three challenging NLP tasks. These tasks form together an extrinsic evaluation setup and go beyond the benchmarks that we have used to develop the respective methods.

We have verified our entity linking approach on two extrinsic tasks: argumenta-

tive reasoning and fact verification. We have shown that our entity linker offers an easy way to annotate an input text with Wikidata entities. For argumentative reasoning, we showed that the Wikidata entity information is helpful for the input that mentions several entities. At the time of the experiment, we were the first to evaluate external knowledge in the context of the argumentative reasoning comprehension dataset of Habernal et al. (2018a). For fact verification, our entity linker has improved the retrieval of the ground truth information by 14 % compared to the previous baseline.

We have shown on the TriviaQA *wiki* and *web* datasets that our graph networks based semantic parser (ExpGGNN) offers a way to use knowledge base interpretation for the text comprehension task. Furthermore, we have developed an interactive evaluation interface that enables a user to explore the linking and semantic parsing models that were developed in this work.

We have answered **RQ3** by presenting experiments across three further NLP tasks beyond only knowledge base question answering: argumentative reasoning, fact verification and text comprehension. We have demonstrated that our developed methods make it possible to link the text to the knowledge base or to parse the text into a complete grounded semantic representation in the context of these applications, which include different domains. The transfer of our methods to the new applications required little implementation costs and resulted in an improved performance across the tested tasks.

Limitations and reflection

The overview of our answers to the formulated research questions would be incomplete with a note on inherent limitations of the knowledge base methods, such as grounded semantic parsing, and consequently of the models developed in this thesis.

The knowledge base linking methods for relation extraction and entity linking (see Chapter 3) are limited by the schema that they inherent from the knowledge base. Wikidata is constructed collaboratively and gradually, which creates two types of coverage gaps: (1) the concepts and relation types should fulfill the notability principle[1] before they are added and some are deemed not notable enough[2]; (2) some topics are simply not yet added to Wikidata. Entities and relation types missing from the Wikidata schema cannot be processed by the knowledge base linking methods, which is a limitation of the approaches explored in this work. A possible solution moving forward is to consider unsupervised methods (Bouraoui et al., 2018) or turn to automatically constructed knowledge bases, such as NELL (Mitchell et al., 2015), that are nosier, but have higher coverage than Wikidata. Relation types that are part of the schema, but are

[1] Wikidata follows the same guidelines as Wikipedia: `https://en.wikipedia.org/wiki/Wikipedia:Notability`

[2] The decision on notability is not always clear cut and there are documented cases of bias: `https://en.wikipedia.org/wiki/Criticism_of_Wikipedia#Notability_of_article_topics`

not yet frequently used throughout the knowledge base form the long-tail. Zhang et al. (2019) focus on long-tail relation extraction by transferring information from frequent relations to the tail of the distribution.

Grounded semantic parsing combines a knowledge base schema with first-order logic to define semantic representations (see Section 2.2.2). Hence, the aforementioned limitations of Wikidata apply to grounded semantic parsing in the same way. Additionally, first-order logic imposes constraints on the expressivity of semantic representations. First-order logic can encode factual statements with co-reference, quantification and negation that are either true or false in an external world model (a knowledge base in our case). It is much harder to encode hypothetical statements with first-order-logic-based approaches, although it is possible to do that to a certain extent using the logic of possible worlds (see van Eijck and Unger, 2010, Chapter 12). Encoding statements with probabilities would require to adopt probabilistic logic (Beltagy et al., 2014) and to combine it with neural methods (Manhaeve et al., 2018), which would in turn require a probabilistic knowledge base (Borgwardt et al., 2018).

In this thesis, we have applied first-order-logic-based semantic parsing methods grounded in Wikidata to single sentences, such as a question in the question answering setup. Application of the same methods to longer documents would require a formalized model of discourse such as the Discourse Representation Theory (Kamp and Reyle, 1996). The relation extraction and entity linking methods that we have developed, on the other hand, can be readily used to link relations and entity mentions in documents. For instance, Yao et al. (2019) apply our relation extraction method on a newly created DocRED documents dataset.

The main application of our semantic parser is the knowledge base question answering task. The semantic parser is limited by the expressivity of the first-order logic representations that it produces and by the Wikidata coverage. It is possible to approach question answering with a fully continuous perspective and without constructing a structured meaning representation grounded in a knowledge base. The promise of these approaches is a fully differentiable architecture that does not require special representation construction procedures (such as in Section 4.2). For instance, Dong et al. (2015) and Jain (2016) construct vector representations for input questions and for candidate answers to compare them directly, though they still retrieve the final answer from a knowledge base. An alternative approach to question answering is to rely on textual source documents instead of a knowledge base. Document question answering methods, which we have discussed in Section 4.5, often employ an encoder-decoder architecture to generate answers based on an input question and source documents (Wang et al., 2018; Seonwoo et al., 2020). However, the recent work has recognized the advantages of a structured knowledge base and has turned to a combination of knowledge bases with documents for question answering (Sun et al., 2018, 2019) and has also adopted GNNs (Feng et al., 2020).

After our investigations and the insights gained in this thesis, we see that the meaning disambiguation methods are constantly advancing. However, it is still challenging to construct a fully unambiguous and complete meaning representation of a single utterance. We have shown (1) that it is crucial to jointly process and disambiguate individual components of the semantic representations; (2) that it is important to include the external knowledge into the interpretation of language, as it might be hard or impossible to learn real world information just from the text; (3) and that the way forward is to combine the semantic processing pipelines into end-to-end methods, but with explicit meaning representations, which explain the automatic system's interpretation. Returning to our view on semantic parsing, which we have sketched in Chapter 1, we call such approaches a *hybrid* model that uses symbolic representations and processes them with continuous vectors.

Automatically understanding the language meaning is the ultimate goal of NLP. Yet, even defining *understanding* and *meaning* is challenging, as humans rely on a seemingly endless situational context and previous experience to encode and decode the meaning of language expressions. In this dissertation, we assume that the meaning of language lies somewhere beyond the language itself and that the language that we observe in textual or other form can only be interpreted when grounded in our knowledge and in our experience (Bender and Koller, 2020). With our work, we add to the growing palette of methods that ground the text in external resources and other modalities. Thereby, we humbly contribute to the grand problem of symbol grounding (Harnad, 1990).

6.2 Impact of the Contributions

We have made a significant contribution to the areas of grounded semantic parsing, relation extraction and entity linking with our work. The publications included in this thesis have been cited together over 230 times[3]. In particular, our publication on context-aware relation extraction (Sorokin and Gurevych, 2017a) and on semantic parsing with GGNNs (Sorokin and Gurevych, 2018a) have received much attention.

Follow-up work to our relation extraction paper (Sorokin and Gurevych, 2017a) Our relation extraction model was directly used by Petroni et al. (2019) as a point of comparison for large language models to test their knowledge base capabilities. Zhu et al. (2019) build upon our model and use GNNs for sentence-level relation extraction. Our Wikidata dataset for relation extraction was used by Zhang et al. (2018) to evaluate their Attention-Based Capsule Networks architecture. Trisedya et al. (2019) extend our dataset construction procedure to collect more training data with Wikidata. A number of publications introduced a new task based

[3] At the time of writing and according to Google Scholar: `https://scholar.google.com/citations?user=PRX-B-oAAAAJ`

on our relation extraction task formulation: Spala et al. (2020) describe Subtask 3: Relation Extraction of SemEval-2020 Task 6 on definition extraction from free text; Yao et al. (2019) presented the DocRED dataset for document-level relation extraction and use our context-aware relation extraction model as a baseline. The later work on the DocRED dataset continued to use our context-aware model as a baseline (Li et al., 2020b; Wang et al., 2020). The code released with our paper (Sorokin and Gurevych, 2017a) was starred 273 times and forked more than 70 times.[4]

Follow-up work to our GGNNs semantic parsing paper (Sorokin and Gurevych, 2018a) Our publication on GGNNs for grounded semantic parsing and knowledge base question answering is frequently cited in the context of applying GGNNs to question answering (Chen et al., 2019; Vakulenko et al., 2019) and to other tasks (Bogin et al., 2019a,b), as well as a reference for successful application of knowledge bases in NLP (Moussallem et al., 2019). Our method for step-by-step graph generation was adopted by Wu and Zhang (2020). The code released with our paper (Sorokin and Gurevych, 2018a) was starred 136 times and forked more than 27 times.[5]

Moreover, our publications in collaboration with other researchers on automatic fact checking (Hanselowski et al., 2018) and on argument reasoning (Botschen et al., 2018b) have been used in follow-up works on the respective tasks. Our document retrieval and evidence ranking from Hanselowski et al. (2018) demonstrated the best result on the FEVER shared task and were used both as a baseline (Thorne et al., 2019; Chernyavskiy and Ilvovsky, 2019) and as a part of a new system (Zhou et al., 2019). Our model for argumentative reasoning enhanced with entity and frame knowledge from Botschen et al. (2018b) was used as a baseline for probing neural network comprehension (Niven and Kao, 2019).

Finally, our work on entity linking (Sorokin and Gurevych, 2018b) has sparked more research on linking entities in questions and short noisy data. Li et al. (2020a) have recently proposed ELQ, which uses additional pre-training for improved performance and a faster architecture, while Banerjee et al. (2020) considered a larger dataset and also improved the model speed. Both works use our VCG entity linker as the baseline and follow our evaluation setup.

6.3 Outlook

Each research question bears a lot of potential for future research. In the following, we list a subjective overview of the most crucial research directions for future work. We summarize the future directions in the updated thesis topic diagram

[4] At the time of writing and according to GitHub: `https://github.com/UKPLab/emnlp2017-relation-extraction/`

[5] At the time of writing and according to GitHub: `https://github.com/UKPLab/coling2018-graph-neural-networks-question-answering`

Figure 6.2: An outlook of further concepts, methods and applications (the outer orbit) and their connections to the topics of this theses (the inner orbit).

in Figure 6.2: the inner orbit shows the topics we have discussed so far and the outer orbit includes new directions.

ConceptNet Weissenborn et al. (2018a) and Feng et al. (2020) use the ConceptNet semantic graph to enrich input documents with additional information. ConceptNet is a graph database that combines lexical information about word usage (e.g. synonyms and hypernyms) and world knowledge (e.g. a cat has four legs).[6] It can be used as an additional source of interpretation for semantic representations along Wikidata to increase the coverage of a grounded semantic parser. Most notably, ConceptNet includes causality information that is lacking from Wikidata,

[6] `http://conceptnet.io`

such as 'fire is capable of *burn houses*'.[7] Causality information makes ConceptNet a useful resource for grounded natural language inference. To integrate ConceptNet, we propose to adapt the definition of the grounded semantic language in Section 2.2.2 to include ConceptNet concepts and predicates.

Visual grounding Natural language understanding is tied to our physical environment (Forbes and Choi, 2017). Knowledge bases offer a factual grounding and interpretation domain for texts (see Section 2.2.2), but grounding texts in images can add another interpretation layer, which includes visual properties such as color or relative positions of two objects (Conser et al., 2019). Positions, size comparison and actions are hard to encode as knowledge base facts, but can be easily demonstrated with an image (Beinborn et al., 2018). This advantage of visual representations can be combined with structured semantic representations, as was shown in Botschen et al. (2018a) for frame semantic parsing. Mousselly-Sergieh et al. (2018) have constructed visual embeddings for Wikidata entities using the images from the linked Wikipedia pages. To advance into the direction of visual grounding, one can add image-derived vectors to the entity representations in our models.

Unsupervised methods The coverage of existing semantic annotation schemes can be increased via unsupervised induction of new semantic relations. The unsupervised methods induce new semantic units that can be forced to fit the chosen semantic formalism. For instance, Titov and Klementiev (2011) induce binary predicates to produce logic-based representations, whereas Modi et al. (2012) decompose this process into induction of predicate frames and of frame specific role sets. Kočiský et al. (2016) combine unsupervised methods with in-domain supervision from a small amount of training data to improve closed-domain semantic parsing.

Contextual representations Recent advances in the word embeddings and pre-trained language models make use of the surrounding context to dynamically produce encodings for the input words and characters. Such methods have been effective on a growing number of semantic NLP tasks (Kovaleva et al., 2019). We have already conducted first preliminary experiments with the ELMO embeddings (Peters et al., 2018) for grounded semantic parsing in Section 4.4. A thorough follow-up investigation on using the pre-trained BERT models (Devlin et al., 2019a) for grounded semantic parsing should be the next step. The main challenge for the experiments with BERT would be the integration of the pre-trained Transformer architecture with the GNNs.

Concept maps Grounded semantic representations can help with summarization in the form of navigation structures or concept maps (Liu et al., 2015). Concept maps provide a user with an overview of the text content in a form of a graph of

[7] `http://conceptnet.io/a/[/r/CapableOf/,/c/en/fire/,/c/en/burn_houses/]`

entities and relations that are described in the text. This allows the user to explore elements of the structure by interactively navigating them. Usually, ungrounded extraction methods are used to create concept maps (Falke and Gurevych, 2017). We suggest to explore the usage of grounded semantic parsing to construct automatic structured summaries of natural texts. One of the main challenges of constructing concept maps is the identification of concept mentions and unification of the mentions that refer to the same underlying concept (Falke and Gurevych, 2019). The Wikidata entity linker alleviates this problem by disambiguating each mention of the same concept to the same unique Wikidata identifier. It is also reasonable to combine the grounded semantic parsing methods with information extraction approaches. First, a concept map of linked Wikidata entities and relations could be created and then enriched with further new concepts extracted from the text.

Language modeling The unsupervised pre-training techniques rely on the language modeling objective: predicting the next word in a document, the masked word in a context or a similar task (Peters et al., 2018; Devlin et al., 2019b). Better language models can lead to better pre-trained blocks for other NLP tasks. Yet, encoding factual information might be difficult for language models that are learned only from text. Petroni et al. (2019) have compared the relation extraction model that we have presented in Section 3.1 against the existing language models and have shown that language models cannot effectively capture factual relations. Logan et al. (2019) have used a distant supervision annotating approach, similar to the one we described in Section 3.1.2, to create a language modeling task using Wikidata. We propose to combine the language modeling objective on a distantly supervised dataset with the question answering objective to train the next GNN architecture.

Community question answering We have already applied our grounded semantic parser to knowledge base question answering and text comprehension. It can be further expanded to community question answering. Community question answering is concerned with questions that are usually posed in an open form and do not presuppose a single correct answer but rather request for more information on a particular subject. For instance, the question 'How can I travel between Germany and Scandinavia?' might be answered with different route suggestions or a discussion of advantages and disadvantages of a particular way of travel. The community question answering methods commonly work with a pool of available answers (often extracted from an online platform) and rank them by relevance for the given input question (Rücklé and Gurevych, 2017). We suggest to leverage the ability of our methods to link the input question and possible answers to the knowledge base. Wikidata semantic parsing can recognize the entities Germany and Scandinavia in the question and the relations between them. It can be used to prioritize the answers that cover the same entities as the question or the entities related to them in Wikidata, such as the cities in Germany

and in the Scandinavian countries. The questions and answers in the community question answering setting tend to contain multiple sentences. Therefore, the main challenge to apply grounded semantic parsing would be to expand it to processing and merging semantic graphs of multiple sentences.

Multilingual semantic parsing Wikidata is an inherently multi-lingual project (Vrandečić and Krötzsch, 2014). From the beginning, it was used as an intermediary to link Wikipedia pages about the same concepts in different languages. Wikidata entity and relation labels are available in many languages and the weak training procedure that we have defined in Section 4.4 is not tied to one language. That leaves a dataset of question-answer pairs as the only requirement to transfer our ExpGGNN architecture to a new language. Furthermore, it might be beneficial to learn the joint model for multiple languages by introducing universal and language specific blocks in the neural architecture. A joint multilingual learning setup has been proven effective for language modeling (Wu and Dredze, 2019) and semantic inference tasks (Conneau et al., 2018).

Presupposition verification In this thesis, we have discussed fact verification as a possible application for linking methods and knowledge bases. A similar use case arises as a subtask in question answering. A question about factual information often presupposes some of the other facts to be correct. For example, at the time of composing the manuscript an online search for "Who is the king of France?" was returning the name of Louis Alphonse, Duke of Anjou. The search results also include information on the Kingdom of France. This information, albeit relevant, is neither the correct nor the most appropriate answer to the given question. A question answering system should recognize that such a question has a false presupposition, namely that the France is a kingdom and that a king of France exists. The correct and the most relevant first response would be a negation of the presupposition: "France has no king.". The identification and verification of presuppositions for input questions would make question answering more robust and intelligent.

List of Figures

List of Tables

List of Abbreviations

AMR abstract meaning representations 21, 22, 23, 24, 26, 117, 179

BiLSTM bidirectional LSTM 97

CNN Convolutional Neural Network 59, 113, 114, 115, 120, 122

DCNN Dilated Convolutional Neural Networks 72, 75, 113, 114, 115, 119, 120, 122, 132, 133, 180

ESIM Enhanced Sequential Inference Model 97, 98

ExpGGNN Gated Graph Neural Network on expanded graphs 7, 120, 121, 122, 123, 124, 125, 126, 127, 128, 131, 132, 133, 134, 135, 137, 138, 144, 145, 146, 147, 148, 149, 153, 169, 170, 177, 183

GCN Graph Convolutional Network 34, 36, 38, 40

GGNN Gated Graph Neural Network v, vii, 9, 34, 36, 38, 39, 40, 41, 42, 102, 105, 112, 115, 116, 117, 118, 119, 120, 121, 122, 124, 125, 126, 127, 128, 132, 136, 148, 149, 166, 167, 168, 169, 172, 173, 179, 180

GNN Graph Neural Network v, vii, 5, 6, 7, 10, 13, 14, 33, 34, 35, 36, 38, 42, 105, 115, 120, 121, 124, 125, 128, 149, 156, 169, 171, 172, 175, 176, 180

LSTM Long Short-Term Memory 56, 57, 58, 59, 60, 62, 89, 141, 142, 143, 144

NER Named Entity Recognition 55, 66, 71, 72, 113

NLP natural langauge processing 1, 2, 3, 4, 5, 6, 7, 8, 10, 14, 16, 17, 18, 19, 20, 22, 26, 27, 30, 33, 34, 41, 42, 43, 44, 45, 53, 65, 75, 84, 85, 87, 100, 115, 130, 163, 165, 167, 169, 170, 172, 173, 175, 176, 179

R-GCN Relational Graph Convolutional Network 36, 37, 39, 40, 41

RNN Recurrent Neural Network 8, 57, 142, 143

VCG Variable Context Granularity iii, 7, 72, 73, 77, 80, 81, 82, 83, 84, 85, 88, 94, 95, 96, 98, 168, 173, 180, 183

Bibliography

Simon Ahrendt and Vera Demberg: 'Improving event prediction by representing script participants', in: *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pp. 546–551, Association for Computational Linguistics, San Diego, USA, 2016, Online: http://aclweb.org/anthology/N16-1067.

Gabor Angeli and Christopher D. Manning: 'NaturalLI: Natural Logic Inference for Common Sense Reasoning', in: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 534–545, Association for Computational Linguistics, Doha, Qatar, October 2014, Online: https://www.aclweb.org/anthology/D14-1059.

Yoav Artzi and Luke Zettlemoyer: 'Bootstrapping Semantic Parsers from Conversations', in: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 421–432, Association for Computational Linguistics, Edinburgh, Scotland, UK, July 2011, Online: https://www.aclweb.org/anthology/D11-1039.

Yoav Artzi and Luke Zettlemoyer: 'Weakly Supervised Learning of Semantic Parsers for Mapping Instructions to Actions', *Transactions of the Association for Computational Linguistics* 1: 49–62, 2013, Online: https://www.aclweb.org/anthology/Q13-1005.

Michael Azmy, Peng Shi, Jimmy Lin, and Ihab Ilyas: 'Farewell Freebase: Migrating the SimpleQuestions Dataset to DBpedia', in: *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*, pp. 2093–2103, Association for Computational Linguistics, Santa Fe, New Mexico, USA, August 2018, Online: https://www.aclweb.org/anthology/C18-1178.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe: 'The Berkeley FrameNet Project', in: *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (Volume 1)*, pp. 86–90, Stroudsburg, PA, USA, 1998, Online: https://www.aclweb.org/anthology/C98-1013.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider: 'Abstract Meaning Representation for Sembanking', in: *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pp. 178–186, Association for Computational Linguistics, Sofia, Bulgaria, August 2013, Online: https://www.aclweb.org/anthology/W13-2322.

Debayan Banerjee, Debanjan Chaudhuri, Mohnish Dubey, and Jens Lehmann:

'PNEL: Pointer Network based End-To-End Entity Linking over Knowledge Graphs', in: *International Semantic Web Conference*, pp. 21–38, Springer, 2020.

Junwei Bao, Nan Duan, Zhao Yan, Ming Zhou, and Tiejun Zhao: 'Constraint-Based Question Answering with Knowledge Graph', in: *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*, pp. 2503–2514, Osaka, Japan, 2016.

Peter Battaglia, Jessica Blake Chandler Hamrick, Victor Bapst, Alvaro Sanchez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andy Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Jayne Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu: 'Relational inductive biases, deep learning, and graph networks', *arXiv* 2018, Online: https://arxiv.org/pdf/1806.01261.pdf.

Daniel Beck, Gholamreza Haffari, and Trevor Cohn: 'Graph-to-Sequence Learning using Gated Graph Neural Networks', in: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 273–283, Association for Computational Linguistics, Melbourne, Australia, July 2018, Online: https://www.aclweb.org/anthology/P18-1026.

Lisa Beinborn, Teresa Botschen, and Iryna Gurevych: 'Multimodal Grounding for Language Processing', in: *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*, pp. 2325–2339, Association for Computational Linguistics, Santa Fe, New Mexico, USA, August 2018, Online: https://www.aclweb.org/anthology/C18-1197.

Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder: 'Adversarial training for multi-context joint entity and relation extraction', in: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2830–2836, Association for Computational Linguistics, Brussels, Belgium, October-November 2018, Online: https://www.aclweb.org/anthology/D18-1307.

Islam Beltagy, Katrin Erk, and Raymond Mooney: 'Semantic Parsing using Distributional Semantics and Probabilistic Logic', in: *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, pp. 7–11, Association for Computational Linguistics, Baltimore, MD, June 2014, Online: https://www.aclweb.org/anthology/W14-2402.

Emily M. Bender and Alexander Koller: 'Climbing towards NLU: On Meaning, Form, and Understanding in the Age of Data', in: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, p. to appear, Association for Computational Linguistics, 2020.

Luisa Bentivogli, Pamela Forner, Claudio Giuliano, Alessandro Marchetti, Emanuele Pianta, and Kateryna Tymoshenko: 'Extending English ACE 2005

corpus annotation with ground-truth links to Wikipedia', in: *Proceedings of the 2nd Workshop on Collaboratively Constructed Semantic Resources at the 23rd International Conference on Computational Linguistics (COLING)*, pp. 19–26, Beijing, China, 2010, Online: http://www.aclweb.org/anthology/W10-3503.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang: 'Semantic Parsing on Freebase from Question-Answer Pairs', in: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1533–1544, Seattle, WA, USA, 2013, Online: http://www.aclweb.org/anthology/D13-1160.

Jonathan Berant and Percy Liang: 'Semantic Parsing via Paraphrasing', in: *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1415–1425, Baltimore, MD, USA, 2014.

Yonatan Bisk, Ari Holtzman, Jesse Thomason, Jacob Andreas, Yoshua Bengio, Joyce Chai, M. Lapata, Angeliki Lazaridou, Jonathan May, Aleksandr Nisnevich, Nicolas Pinto, and Joseph P. Turian: 'Experience Grounds Language', *ArXiv* 2020.

Ben Bogin, Jonathan Berant, and Matt Gardner: 'Representing Schema Structure with Graph Neural Networks for Text-to-SQL Parsing', in: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4560–4565, Association for Computational Linguistics, Florence, Italy, July 2019a, Online: https://www.aclweb.org/anthology/P19-1448.

Ben Bogin, Matt Gardner, and Jonathan Berant: 'Global Reasoning over Database Structures for Text-to-SQL Parsing', 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)*, pp. 3659–3664, Association for Computational Linguistics, Hong Kong, China, November 2019b, Online: https://www.aclweb.org/anthology/D19-1378.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov: 'Enriching Word Vectors with Subword Information', *Transactions of the Association for Computational Linguistics* 5: 135–146, 2017, Online: https://www.aclweb.org/anthology/Q17-1010.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston: 'Large-scale simple question answering with memory networks', *ArXiv* 2015.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko: 'Translating Embeddings for Modeling Multi-Relational Data', in: *Proceedings of the 26th International Conference on Advances in Neural Information Processing Systems (NIPS)*, pp. 2787–2795, Curran Associates, Inc., Lake Tahoe, NV, USA, 2013.

Stefan Borgwardt, İsmail İlkan Ceylan, and Thomas Lukasiewicz: 'Recent advances in querying probabilistic knowledge bases', in: *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 5420–5426, 2018.

Teresa Botschen, Iryna Gurevych, Jan-Christoph Klie, Hatem Mousselly Sergieh, and Stefan Roth: 'Multimodal Frame Identification with Multilingual Evaluation', in: *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pp. 1481–1491, Association for Computational Linguistics, 2018a.

Teresa Botschen, Hatem Mousselly-Sergieh, and Iryna Gurevych: 'Prediction of Frame-to-Frame Relations in the FrameNet Hierarchy with Frame Embeddings', in: *Proceedings of th 2nd Workshop on Representation Learning for NLP (RepL4NLP)*, pp. 146–156, August 2017.

Teresa Botschen, Daniil Sorokin, and Iryna Gurevych: 'Frame- and Entity-Based Knowledge for Common-Sense Argumentative Reasoning', in: *Proceedings of the 5th Workshop on Argument Mining*, pp. 90–96, Association for Computational Linguistics, Brussels, Belgium, November 2018b, Online: `https://www.aclweb.org/anthology/W18-5211`.

Zied Bouraoui, Shoaib Jameel, and Steven Schockaert: 'Relation Induction in Word Embeddings Revisited', in: *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 1627–1637, Association for Computational Linguistics, Santa Fe, New Mexico, USA, August 2018, Online: `https://www.aclweb.org/anthology/C18-1138`.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning: 'A large annotated corpus for learning natural language inference', in: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 632–642, Association for Computational Linguistics, 2015.

Razvan Bunescu and Marius Pasca: 'Using Encyclopedic Knowledge for Named Entity Disambiguation', in: *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pp. 9–16, Trento, Italy, 2006, Online: `http://aclweb.org/anthology/E06-1002`.

David Carmel, Ming-Wei Chang, Evgeniy Gabrilovich, Bo-June (Paul) Hsu, and Kuansan Wang: 'ERD'14: Entity Recognition and Disambiguation Challenge', in: *ACM SIGIR Forum*, Vol. 48, pp. 63–77, 2014.

Ming-Wei Chang, Bo-June Hsu, Hao Ma, Ricky Loynd, and Kuansan Wang: 'E2E: An End-to-End Entity Linking System for Short and Noisy text', in: *Proceedings of the 4thWorkshop on Making Sense of Microposts co-located with the 23rd International World Wide Web Conference (WWW)*, pp. 62–63, Seoul, Korea, 2014.

Long Chen, Joemon M. Jose, Haitao Yu, Fajie Yuan, and Dell Zhang: 'A Semantic Graph based Topic Model for Question Retrieval in Community Question Answering', in: *Proceedings of the 9th ACM International Conference on Web Search and Data Mining (WSDM)*, pp. 287–296, San Francisco, CA, USA, 2016.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen: 'Enhanced LSTM for Natural Language Inference', in: *Proceedings of the 55th*

Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1657–1668, Association for Computational Linguistics, Vancouver, Canada, July 2017, Online: https://www.aclweb.org/anthology/P17-1152.

Zi-Yuan Chen, Chih-Hung Chang, Yi-Pei Chen, Jijnasa Nayak, and Lun-Wei Ku: 'UHop: An Unrestricted-Hop Relation Extraction Framework for Knowledge-Based Question Answering', 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)*, pp. 345–356, Association for Computational Linguistics, Minneapolis, Minnesota, June 2019, Online: https://www.aclweb.org/anthology/N19-1031.

Jianpeng Cheng, Siva Reddy, Vijay Saraswat, and Mirella Lapata: 'Learning Structured Natural Language Representations for Semantic Parsing', in: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 44–55, Association for Computational Linguistics, Vancouver, Canada, July 2017, Online: https://www.aclweb.org/anthology/P17-1005.

Pengxiang Cheng and Katrin Erk: 'Implicit Argument Prediction with Event Knowledge', in: *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pp. 831–840, Association for Computational Linguistics, 2018.

Anton Chernyavskiy and Dmitry Ilvovsky: 'Extract and Aggregate: A Novel Domain-Independent Approach to Factual Data Verification', in: *Proceedings of the Second Workshop on Fact Extraction and VERification (FEVER)*, pp. 69–78, Association for Computational Linguistics, Hong Kong, China, November 2019, Online: https://www.aclweb.org/anthology/D19-6612.

Kyunghyun Cho, Bart Van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio: 'Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation Kyunghyun', in: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Doha, Qatar, 2014, Online: http://emnlp2014.org/papers/pdf/EMNLP2014179.pdf.

HongSeok Choi and Hyunju Lee: 'GIST at SemEval-2018 Task 12: A network transferring inference knowledge to Argument Reasoning Comprehension task', in: *Proceedings of The 12th International Workshop on Semantic Evaluation*, pp. 773–777, Association for Computational Linguistics, New Orleans, Louisiana, June 2018, Online: https://www.aclweb.org/anthology/S18-1122.

Christopher Clark and Matt Gardner: 'Simple and Effective Multi-Paragraph Reading Comprehension', in: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1, pp. 845–855, 2018.

Stephen Clark and James R. Curran: 'Wide-Coverage Efficient Statistical Parsing

with CCG and Log-Linear Models', *Computational Linguistics* 33 (4): 493–552, 2007, Online: https://www.aclweb.org/anthology/J07-4004.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov: 'XNLI: Evaluating Cross-lingual Sentence Representations', in: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2475–2485, Association for Computational Linguistics, Brussels, Belgium, October-November 2018, Online: https://www.aclweb.org/anthology/D18-1269.

Erik Conser, Kennedy Hahn, Chandler Watson, and Melanie Mitchell: 'Revisiting Visual Grounding', in: *Proceedings of the Second Workshop on Shortcomings in Vision and Language*, pp. 37–46, Association for Computational Linguistics, Minneapolis, Minnesota, June 2019, Online: https://www.aclweb.org/anthology/W19-1804.

Alan Cooper, Robert Reimann, and David Cronin: *About Face 3: The Essentials of Interaction Design*, John Wiley & Sons, 2007.

Silviu Cucerzan: 'Large-Scale Named Entity Disambiguation Based on Wikipedia Data', in: *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 708–716, Prague, Czech Republic, 2007, Online: http://aclweb.org/anthology/D07-1074.

Silviu Cucerzan: 'The MSR System for Entity Linking at TAC 2012', in: *Proceedings of the Text Analysis Conference (TAC)*, pp. 14–15, Gaithersburg, MD, USA, 2012.

Dipanjan Das, Desai Chen, André F. T. Martins, Nathan Schneider, and Noah A. Smith: 'Frame-Semantic Parsing', *Computational Linguistics* 40:1: 9–56, 2014.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova: '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)*, pp. 4171–4186, Association for Computational Linguistics, Minneapolis, Minnesota, June 2019a, Online: https://www.aclweb.org/anthology/N19-1423.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova: '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)*, pp. 4171–4186, Association for Computational Linguistics, Minneapolis, Minnesota, June 2019b, Online: https://www.aclweb.org/anthology/N19-1423.

Dennis Diefenbach, Thomas Tanon, Kamal Singh, and Pierre Maret: 'Question an-

swering benchmarks for Wikidata', in: *In Proceedings of the International Semantic Web Conference (ISWC)*, Vienna, Austria, October 2017.

Li Dong and Mirella Lapata: 'Language to Logical Form with Neural Attention', in: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pp. 33–43, 2016.

Li Dong and Mirella Lapata: 'Coarse-to-Fine Decoding for Neural Semantic Parsing', in: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 731–742, Association for Computational Linguistics, Melbourne, Australia, July 2018, Online: https://www.aclweb.org/anthology/P18-1068.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu: 'Question Answering over Freebase with Multi-Column Convolutional Neural Networks', in: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pp. 260–269, Association for Computational Linguistics, Beijing, China, 2015.

Jan van Eijck and Christina Unger: *Computational Semantics with Functional Programming*, Cambridge University Press, USA, 1st edition, 2010.

Hady Elsahar, Christophe Gravier, and Frederique Laforest: 'Zero-Shot Question Generation from Knowledge Graphs for Unseen Predicates and Entity Types', in: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, Vol. 1, pp. 218–228, 2018.

Tobias Falke and Iryna Gurevych: 'Bringing Structure into Summaries: Crowdsourcing a Benchmark Corpus of Concept Maps', in: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2951–2961, Association for Computational Linguistics, Copenhagen, Denmark, September 2017, Online: https://www.aclweb.org/anthology/D17-1320.

Tobias Falke and Iryna Gurevych: 'Fast Concept Mention Grouping for Concept Map-based Multi-Document Summarization', 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)*, pp. 695–700, Association for Computational Linguistics, Minneapolis, Minnesota, June 2019, Online: https://www.aclweb.org/anthology/N19-1074.

Federico Fancellu, Siva Reddy, Adam Lopez, and Bonnie Webber: 'Universal Dependencies to Logical Form with Negation Scope', in: *Proceedings of the Workshop Computational Semantics Beyond Events and Roles*, pp. 22–32, Association for Computational Linguistics, Valencia, Spain, April 2017, Online: https://www.aclweb.org/anthology/W17-1804.

Michael Färber, Basil Ell, Carsten Menne, and Achim Rettinger: 'A Comparative

Survey of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO', *Semantic Web Journal* 1: 1–5, 2015.

Christiane Fellbaum (Ed.): *WordNet: An Electronic Lexical Database*, Language, Speech, and Communication, MIT Press, Cambridge, MA, 1998.

Yanlin Feng, Xinyue Chen, Bill Yuchen Lin, Peifeng Wang, Jun Yan, and Xiang Ren: 'Scalable Multi-Hop Relational Reasoning for Knowledge-Aware Question Answering', in: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1295–1309, Association for Computational Linguistics, Online, November 2020, Online: https://www.aclweb.org/anthology/2020.emnlp-main.99.

Charles J. Fillmore: 'Frame semantics and the nature of language', *Annals of the New York Academy of Sciences* 280: 20–32, 1976.

Maxwell Forbes and Yejin Choi: 'Verb Physics: Relative Physical Knowledge of Actions and Objects', in: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 266–276, Association for Computational Linguistics, Vancouver, Canada, July 2017, Online: https://www.aclweb.org/anthology/P17-1025.

Matthew Francis-Landau, Greg Durrett, and Dan Klein: 'Capturing Semantic Similarity for Entity Linking with Convolutional Neural Networks', in: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1256–1261, Association for Computational Linguistics, San Diego, California, June 2016, Online: https://www.aclweb.org/anthology/N16-1150.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer: 'AllenNLP: A Deep Semantic Natural Language Processing Platform', in: *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pp. 1–6, Association for Computational Linguistics, Melbourne, Australia, July 2018, Online: https://www.aclweb.org/anthology/W18-2501.

Abbas Ghaddar and Phillippe Langlais: 'WiNER: A Wikipedia Annotated Corpus for Named Entity Recognition', in: *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 413–422, Asian Federation of Natural Language Processing, Taipei, Taiwan, November 2017, Online: https://www.aclweb.org/anthology/I17-1042.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl: 'Neural Message Passing for Quantum Chemistry', in: *Proceedings of the International Conference on Machine Learning (ICML)*, p. 1263–1272, 2017.

Max Glockner, Vered Shwartz, and Yoav Goldberg: 'Breaking NLI Systems with Sentences that Require Simple Lexical Inferences', in: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Pa-*

pers), pp. 650–655, Association for Computational Linguistics, Melbourne, Australia, July 2018, Online: https://www.aclweb.org/anthology/P18-2103.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville: *Deep Learning*, The MIT Press, 2016.

Bert Green, Alice Wolf, Carol Chomsky, and Kenneth Laughery: 'Baseball: An automatic Question Answerer', in: *Proceedings of Western Computing Conference*, pp. 219–224, 1961.

H. Paul Grice: 'Logic and Conversation', in Peter Cole and Jerry L. Morgan (Eds.): *Syntax and Semantics: Vol. 3: Speech Acts*, pp. 41–58, Academic Press, New York, 1975.

Stephen Guo, Ming-Wei Chang, and Emre Kiciman: 'To Link or Not to Link? A Study on End-to-End Tweet Entity Linking', in: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pp. 1020–1030, Atlanta, GA, USA, 2013a, Online: http://www.aclweb.org/anthology/N13-1122.

Yuhang Guo, Bing Qin, Ting Liu, and Sheng Li: 'Microblog entity linking by leveraging extra posts', in: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 863–868, 2013b, Online: http://www.aclweb.org/anthology/D13-1085.

Iryna Gurevych, Judith Eckle-Kohler, and Michael Matuschek: 'Linked Lexical Knowledge Bases: Foundations and Applications', *Synthesis Lectures on Human Language Technologies* 9 (3): 1–146, 2016, Online: https://doi.org/10.2200/S00717ED1V01Y201605HLT034.

Ivan Habernal, Henning Wachsmuth, Iryna Gurevych, and Benno Stein: 'The Argument Reasoning Comprehension Task: Identification and Reconstruction of Implicit Warrants', in: *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pp. 1930–1940, Association for Computational Linguistics, New Orleans, Louisiana, June 2018a, Online: https://www.aclweb.org/anthology/N18-1175.

Ivan Habernal, Henning Wachsmuth, Iryna Gurevych, and Benno Stein: 'SemEval-2018 Task 12: The Argument Reasoning Comprehension Task', in: *Proceedings of The 12th International Workshop on Semantic Evaluation*, pp. 763–772, Association for Computational Linguistics, New Orleans, Louisiana, June 2018b, Online: https://www.aclweb.org/anthology/S18-1121.

Xianpei Han and Le Sun: 'An entity-topic model for entity linking', in: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 105–115, Jeju Island, Korea, 2012, Online: http://www.aclweb.org/anthology/D12-1010.

Andreas Hanselowski, Hao Zhang, Zile Li, Daniil Sorokin, Benjamin Schiller,

Claudia Schulz, and Iryna Gurevych: 'UKP-Athene: Multi-Sentence Textual Entailment for Claim Verification', in: *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pp. 103–108, Association for Computational Linguistics, Brussels, Belgium, November 2018, Online: https://www.aclweb.org/anthology/W18-5516.

Stevan Harnad: 'The symbol grounding problem', *Physica D: Nonlinear Phenomena* 42 (1): 335 – 346, 1990.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom: 'Teaching machines to read and comprehend', in: *Advances in Neural Information Processing Systems*, pp. 1693–1701, 2015.

Daniel Hewlett, Alexandre Lacoste, Llion Jones, Illia Polosukhin, Andrew Fandrianto, Jay Han, Matthew Kelcey, and David Berthelot: 'WikiReading: A Novel Large-scale Language Understanding Task over Wikipedia', in: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1535–1545, Association for Computational Linguistics, Berlin, Germany, August 2016, Online: https://www.aclweb.org/anthology/P16-1145.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston: 'The goldilocks principle: Reading children's books with explicit memory representations', in: *Proceedings of the International Conference on Learning Representations*, San Juan, Puerto Rico, 2016.

Sepp Hochreiter and Jürgen Schmidhuber: 'Long short-term memory', *Neural Computation* 9 (8): 1–32, 1997.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum: 'Robust Disambiguation of Named Entities in Text', in: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 782–792, Edinburgh, Scotland, UK, 2011, Online: http://www.aclweb.org/anthology/D11-1072.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld: 'Knowledge-based weak supervision for information extraction of overlapping relations', in: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, pp. 541–550, Portland, Oregon, USA, 2011.

Sarthak Jain: 'Question Answering over Knowledge Base using Factual Memory Networks', in: *Proceedings of the NAACL Student Research Workshop*, pp. 109–115, San Diego, CA, USA, 2016.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer: 'TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension', in: *Proceedings of the 55th Annual Meeting of the Association for Computational*

Linguistics (Volume 1: Long Papers), pp. 1601–1611, Association for Computational Linguistics, 2017, Online: http://aclweb.org/anthology/P17-1147.

Lucie-Aimée Kaffee, Hady Elsahar, Pavlos Vougiouklis, Christophe Gravier, Frederique Laforest, Jonathon Hare, and Elena Simperl: 'Learning to Generate Wikipedia Summaries for Underserved Languages from Wikidata', in: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pp. 640–645, Association for Computational Linguistics, New Orleans, Louisiana, June 2018, Online: https://www.aclweb.org/anthology/N18-2101.

Hans Kamp and Uwe Reyle: *A calculus for first order Discourse Representation Structures*, Vol. 5, 1996.

Divyansh Kaushik and Zachary C. Lipton: 'How Much Reading Does Reading Comprehension Require? A Critical Investigation of Popular Benchmarks', in: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 5010–5015, Association for Computational Linguistics, Brussels, Belgium, October-November 2018, Online: https://www.aclweb.org/anthology/D18-1546.

Daniel Khashabi, Tushar Khot, Ashish Sabharwal, and Dan Roth: 'Question Answering as Global Reasoning over Semantic Abstractions', in: *Proceedings of the 32nd Conference of Association for the Advancement of Artificial Intelligence (AAAI)*, pp. 1905–1914, AAAI Press, 2018.

Diederik Kingma and Jimmy Ba: 'Adam: A Method for Stochastic Optimization', *ArXiv* 2014.

Thomas N. Kipf and Max Welling: 'Semi-Supervised Classification with Graph Convolutional Networks', in: *In Proceedings of the 5th International Conference on Learning Representations (ICLR)*, pp. 1–14, Toulon, France, 2017.

Tomáš Kočiský, Gábor Melis, Edward Grefenstette, Chris Dyer, Wang Ling, Phil Blunsom, and Karl Moritz Hermann: 'Semantic Parsing with Semi-Supervised Sequential Autoencoders', in: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1078–1087, Association for Computational Linguistics, Austin, Texas, November 2016, Online: https://www.aclweb.org/anthology/D16-1116.

Lev Konstantinovskiy, Oliver Price, Mevan Babakar, and Arkaitz Zubiaga: 'Towards Automated Factchecking: Developing an Annotation Schema and Benchmark for Consistent Automated Claim Detection', *arXiv* 2018.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky: 'Revealing the Dark Secrets of BERT', 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)*, pp. 4365–4374, Association for

Computational Linguistics, Hong Kong, China, November 2019, Online: https://www.aclweb.org/anthology/D19-1445.

Ashwini Jaya Kumar, Christoph Schmidt, and Joachim Köhler: 'A knowledge graph based speech interface for question answering systems', *Speech Communication* 92: 1–12, 2017.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc V. Le, and Slav Petrov: 'Natural Questions: a Benchmark for Question Answering Research', *Transactions of the Association of Computational Linguistics* 2019.

Brenden Lake and Marco Baroni: 'Still not systematic after all these years: On the compositional skills of sequence-to-sequence recurrent networks', in: *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

Friedrich Wilhelm Levi: *Finite geometrical systems: six public lectues delivered in February, 1940, at the University of Calcutta*, The University of Calcutta, 1942.

Belinda Z. Li, Sewon Min, Srinivasan Iyer, Yashar Mehdad, and Wen-tau Yih: 'Efficient One-Pass End-to-End Entity Linking for Questions', in: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6433–6441, Association for Computational Linguistics, Online, November 2020a, Online: https://www.aclweb.org/anthology/2020.emnlp-main.522.

Bo Li, Wei Ye, Zhonghao Sheng, Rui Xie, Xiangyu Xi, and Shikun Zhang: 'Graph Enhanced Dual Attention Network for Document-Level Relation Extraction', in: *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 1551–1560, International Committee on Computational Linguistics, Barcelona, Spain (Online), December 2020b, Online: https://www.aclweb.org/anthology/2020.coling-main.136.

Qi Li, Heng Ji, and Liang Huang: 'Joint Event Extraction via Structured Prediction with Global Features', in: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 73–82, Sofia, Bulgaria, 2013, Online: http://www.aclweb.org/anthology/P13-1008.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel: 'Gated Graph Sequence Neural Networks', in: *In Proceedings of the 4th International Conference on Learning Representations (ICLR)*, pp. 1–19, San Juan, Puerto Rico, 2016.

Percy Liang: 'Learning Executable Semantic Parsers for Natural Language Understanding', *Communications of the ACM* 59 (9): 68–76, 2016.

Percy Liang, Michael Jordan, and Dan Klein: 'Learning Dependency-Based Compositional Semantics', in: *Proceedings of the 49th Annual Meeting of the Association*

for Computational Linguistics: Human Language Technologies, pp. 590–599, Association for Computational Linguistics, Portland, Oregon, USA, June 2011, Online: https://www.aclweb.org/anthology/P11-1060.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun: 'Neural Relation Extraction with Selective Attention over Instances', in: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 2124–2133, Berlin, Germany, 2016.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah a. Smith: 'Toward Abstractive Summarization Using Semantic Representations', in: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1076–1085, 2015.

Robert Logan, Nelson F. Liu, Matthew E. Peters, Matt Gardner, and Sameer Singh: 'Barack's Wife Hillary: Using Knowledge Graphs for Fact-Aware Language Modeling', in: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5962–5971, Association for Computational Linguistics, Florence, Italy, July 2019, Online: https://www.aclweb.org/anthology/P19-1598.

Vanessa Lopez, Pierpaolo Tommasi, Spyros Kotoulas, and Jiewen Wu: 'QuerioDALI: Question answering over dynamic and linked knowledge graphs', in: *The Semantic Web - 15th International Semantic Web Conference (ISWC 2016)*, pp. 363–382, Springer International Publishing, Kobe, Japan, 2016.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie: 'Joint Named Entity Recognition and Disambiguation', in: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Lisbon, Portugal, 2015.

Kangqi Luo, Fengli Lin, Xusheng Luo, and Kenny Zhu: 'Knowledge Base Question Answering via Encoding of Complex Query Graphs', in: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2185–2194, Association for Computational Linguistics, Brussels, Belgium, October-November 2018, Online: https://www.aclweb.org/anthology/D18-1242.

Thang Luong, Hieu Pham, and Christopher D. Manning: 'Effective Approaches to Attention-based Neural Machine Translation', in: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1412–1421, Association for Computational Linguistics, 2015, Online: http://aclweb.org/anthology/D15-1166.

Laurens van der Maaten and Geoffrey Hinton: 'Visualizing High-Dimensional Data using t-SNE', *Journal of Machine Learning Research* 9: 2579–2605, November 2008.

Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester,

and Luc De Raedt: 'Deepproblog: Neural probabilistic logic programming', *Advances in Neural Information Processing Systems* 31: 3749–3759, 2018.

Christopher D. Manning, John Bauer, Jenny Finkel, Steven J Bethard, Mihai Surdeanu, and David McClosky: 'The Stanford CoreNLP Natural Language Processing Toolkit', in: *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics (ACL): System Demonstrations*, pp. 55–60, Baltimore, Maryland, USA, 2014.

Diego Marcheggiani and Ivan Titov: 'Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling', in: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1506–1515, Association for Computational Linguistics, Copenhagen, Denmark, 2017.

Pablo N. Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer: 'DBpedia Spotlight: Shedding Light on the Web of Documents', in: *Proceedings of the 7th International Conference on Semantic Systems (I-Semantics)*, Vol. 95, pp. 1–8, Graz, Austria, 2011.

Todor Mihaylov and Anette Frank: 'Knowledgeable Reader: Enhancing Cloze-Style Reading Comprehension with External Commonsense Knowledge', in: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 821–832, Association for Computational Linguistics, Melbourne, Australia, July 2018, Online: `https://www.aclweb.org/anthology/P18-1076`.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean: 'Efficient Estimation of Word Representations in Vector Space', *ArXiv* 2013.

Alexander Miller, Will Feng, Dhruv Batra, Antoine Bordes, Adam Fisch, Jiasen Lu, Devi Parikh, and Jason Weston: 'ParlAI: A Dialog Research Software Platform', in: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 79–84, Association for Computational Linguistics, Copenhagen, Denmark, September 2017, Online: `https://www.aclweb.org/anthology/D17-2014`.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston: 'Key-Value Memory Networks for Directly Reading Documents', in: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1400–1409, Association for Computational Linguistics, Austin, Texas, November 2016, Online: `https://www.aclweb.org/anthology/D16-1147`.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky: 'Distant supervision for relation extraction without labeled data', in: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pp. 1003–1011, Singapore, 2009.

Tom Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, Justin Bet-

teridge, Andrew Carlson, Bhavana Dalvi Mishra, Matthew Gardner, Bryan Kisiel, Jayant Krishnamurthy, Ni Lao, Kathryn Mazaitis, Thahir Mohamed, Ndapa Nakashole, Emmanouil Platanios, Alan Ritter, Mehdi Samadi, Burr Settles, Richard Wang, Derry Wijaya, Abhinav Gupta, Xinlei Chen, Abulhair Saparov, Malcolm Greaves, and Joel Welling: 'Never-ending learning', in: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*, pp. 2302–2310, January 2015.

Makoto Miwa and Mohit Bansal: 'End-to-end Relation Extraction using LSTMs on Sequences and Tree Structures', in: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1105–1116, Berlin, Germany, 2016.

Ashutosh Modi, Ivan Titov, and Alexandre Klementiev: 'Unsupervised Induction of Frame-Semantic Representations', in: *Proccedings of the NAACL-HLT Workshop on the Induction of Linguistic Structure*, pp. 1–7, Montréal, Canada, 2012.

Richard Montague: 'Universal Grammar', in Richmond H. Thomason (Ed.): *Formal Philosophy: Selected Papers of Richard Montague*, pp. 222–247, Yale University Press, New Haven, London, 1974.

Raymond J. Mooney: 'Learning for Semantic Parsing', in Alexander Gelbukh (Ed.): *Computational Linguistics and Intelligent Text Processing*, pp. 311–324, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

Andrea Moro, Alessandro Raganato, and Roberto Navigli: 'Entity Linking meets Word Sense Disambiguation: a Unified Approach', *Transactions of the Association for Computational Linguistics* 2: 231–244, 2014, Online: `https://www.aclweb.org/anthology/Q14-1019`.

Diego Moussallem, Tommaso Soru, and Axel-Cyrille Ngonga Ngomo: 'THOTH: Neural Translation and Enrichment of Knowledge Graphs', in: *International Semantic Web Conference (ISWC)*, pp. 505–522, Springer, 2019.

Hatem Mousselly-Sergieh, Teresa Botschen, Iryna Gurevych, and Stefan Roth: 'A Multimodal Translation-Based Approach for Knowledge Graph Representation Learning', in: *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pp. 225–234, Association for Computational Linguistics, New Orleans, Louisiana, June 2018, Online: `https://www.aclweb.org/anthology/S18-2027`.

Hatem Mousselly-Sergieh and Iryna Gurevych: 'Enriching Wikidata with Frame Semantics', in: *Proceedings of the 5th Workshop on Automated Knowledge Base Construction (AKBC, held in conjunction with NAACL)*, pp. 29–34, 2016.

Timothy Niven and Hung-Yu Kao: 'Probing Neural Network Comprehension of Natural Language Arguments', in: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4658–4664, Association for

Computational Linguistics, Florence, Italy, July 2019, Online: https://www.aclweb.org/anthology/P19-1459.

Martha Palmer, Daniel Gildea, and Nianwen Xue: *Semantic Role Labeling*, Synthesis Lectures on Human La, Morgan & Claypool Publishers, 2010.

Ankur P. Parikh, Hoifung Poon, and Kristina Toutanova: 'Grounded Semantic Parsing for Complex Knowledge Extraction', in: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 756–766, Association for Computational Linguistics, Denver, Colorado, June 2015, Online: https://www.aclweb.org/anthology/N15-1077.

Haoruo Peng, Ming-Wei Chang, and Wen-Tau Yih: 'Maximum Margin Reward Networks for Learning from Explicit and Implicit Supervision', in: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2358–2368, Association for Computational Linguistics, Copenhagen, Denmark, 2017.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning: 'GloVe: Global Vectors for Word Representation', in: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, Doha, Qatar, 2014.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer: '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)*, pp. 2227–2237, Association for Computational Linguistics, New Orleans, Louisiana, June 2018, Online: https://www.aclweb.org/anthology/N18-1202.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller: 'Language Models as Knowledge Bases?', in: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2463–2473, Association for Computational Linguistics, Hong Kong, China, November 2019, Online: https://www.aclweb.org/anthology/D19-1250.

Pranav Rajpurkar, Robin Jia, and Percy Liang: '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)*, pp. 784–789, Association for Computational Linguistics, Melbourne, Australia, July 2018, Online: https://www.aclweb.org/anthology/P18-2124.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang: 'SQuAD: 100,000+ Questions for Machine Comprehension of Text', in: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–

2392, Association for Computational Linguistics, 2016, Online: http://aclweb.org/anthology/D16-1264.

Jef Raskin: *The Humane Interface: New Directions for Designing Interactive Systems*, Addison-Wesley, 2000.

Siva Reddy, Mirella Lapata, and Mark Steedman: 'Large-scale Semantic Parsing without Question-Answer Pairs', *Transactions of the Association for Computational Linguistics* 2: 377–392, 2014, Online: http://aclweb.org/anthology/Q14-1030.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata: 'Transforming Dependency Structures to Logical Forms for Semantic Parsing', *Transactions of the Association for Computational Linguistics* 4: 127–140, 2016, Online: http://aclweb.org/anthology/Q16-1010.

Siva Reddy, Oscar Täckström, Slav Petrov, Mark Steedman, and Mirella Lapata: 'Universal Semantic Parsing', in: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 89–101, Copenhagen, Denmark, 2017.

Nils Reimers and Iryna Gurevych: 'Why Comparing Single Performance Scores Does Not Allow to Draw Conclusions About Machine Learning Approaches', *ArXiv* 2018.

Frank Richter and Manfred Sailer: 'Modeling Typological Markedness in Semantics: The Case of Negative Concord', in Stefan Müller (Ed.): *Proceedings of the 13th International Conference on Head-Driven Phrase Structure Grammar*, CSLI Publications, pp. 305–325, 2006.

Sebastian Riedel, Limin Yao, and Andrew McCallum: 'Modeling Relations and Their Mentions without Labeled Text', in: *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 148–163, Barcelona, Spain, 2010.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni: 'Named Entity Recognition in Tweets: An Experimental Study', in: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1524–1534, Association for Computational Linguistics, Edinburgh, Scotland, UK, July 2011, Online: https://www.aclweb.org/anthology/D11-1141.

Giuseppe Rizzo, Bianca Pereira, Andrea Varga, Marieke Van Erp, and Amparo Elizabeth Cano Basave: 'Lessons learnt from the Named Entity rEcognition and Linking (NEEL) Challenge Series', *Semantic Web* 8 (5): 667–700, 2017.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomas Kocisky, and Phil Blunsom: 'Reasoning about Entailment with Neural Attention', in: *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

Dan Roth and Wen-Tau Yih: 'A linear programming formulation for global inference in natural language tasks', in: *Proceedings of the 8th Conference on Computational Natural Language Learning (CoNLL)*, pp. 1–8, Boston, Massachusetts, USA, 2004.

Jef Rubin and Dana Chisnell: *Handbook of Usability Testing: Howto Plan, Design, and Conduct Effective Tests*, Wiley Publishing, 2nd edition, 2008.

Andreas Rücklé and Iryna Gurevych: 'Representation Learning for Answer Selection with LSTM-Based Importance Weighting', in: *IWCS 2017 — 12th International Conference on Computational Semantics — Short papers*, 2017, Online: `https://www.aclweb.org/anthology/W17-6935`.

Josef Ruppenhofer, Michael Ellsworth, Miriam R.L. Petruck, Christopher R. Johnson, and Jan Scheffczyk: *FrameNet II: Extended Theory and Practice*, International Computer Science Institute, Berkeley, USA, 2016.

Rob A. van der Sandt: 'Presupposition Projection as Anaphora Resolution', *Journal of Semantics* 9 (4): 333–377, 1992.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini: 'The Graph Neural Network Model', *IEEE Transactions on Neural Networks* 20 (1): 61–80, 2009, Online: `http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=4700287`.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling: 'Modeling Relational Data with Graph Convolutional Networks', in Aldo Gangemi, Roberto Navigli, Maria-Esther Vidal, Pascal Hitzler, Raphaël Troncy, Laura Hollink, Tordai Anna, and Mehwish Alam (Eds.): *The Semantic Web*, pp. 593–607, Springer International Publishing, Cham, 2018.

Yeon Seonwoo, Ji-Hoon Kim, Jung-Woo Ha, and Alice Oh: 'Context-Aware Answer Extraction in Question Answering', in: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2418–2428, Association for Computational Linguistics, Online, November 2020, Online: `https://www.aclweb.org/anthology/2020.emnlp-main.189`.

Avirup Sil, Gourab Kundu, Radu Florian, and Wael Hamza: 'Neural Cross-Lingual Entity Linking', in: *Association for the Advancement of Artificial Intelligence (AAAI)*, New Orleans, LA, US, 2018.

Kuldeep Singh, Andreas Both, Dennis Diefenbach, Saedeeh Shekarpour, Didier Cherix, and Christoph Lange: 'Qanary – The Fast Track to Creating a Question Answering System with Linked Data Technology', in Harald Sack, Giuseppe Rizzo, Nadine Steinmetz, Dunja Mladenić, Sören Auer, and Christoph Lange (Eds.): *The Semantic Web*, pp. 183–188, Springer International Publishing, Cham, 2016.

Daniil Sorokin, Daniel Faber, Lars Wolf, and Iryna Gurevych: 'Transforming Knowledge Base Questions into Structured Queries', 2019. Draft.

Daniil Sorokin and Iryna Gurevych: 'Context-Aware Representations for Knowledge Base Relation Extraction', in: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1784–1789, Association for Computational Linguistics, Copenhagen, Denmark, September 2017a, Online: https://www.aclweb.org/anthology/D17-1188.

Daniil Sorokin and Iryna Gurevych: 'End-to-end representation learning for question answering with weak supervision', in Mauro Dragoni, Monika Solanki, and Eva Blomqvist (Eds.): *Semantic Web Challenges — 4th SemWebEval Challenge at ESWC 2017, Portoroz, Slovenia, May 28 – June 1, 2017, Revised Selected Papers*, Communications in Computer and Information Science Vol. 769, pp. 70–83, Springer, 2017b.

Daniil Sorokin and Iryna Gurevych: 'Interactive Instance-based Evaluation of Knowledge Base Question Answering', in: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 114–119, Association for Computational Linguistics, Brussels, Belgium, November 2018c, Online: https://www.aclweb.org/anthology/D18-2020.

Daniil Sorokin and Iryna Gurevych: 'Mixing Context Granularities for Improved Entity Linking on Question Answering Data across Entity Categories', in: *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics (*SEM)*, pp. 65–75, Association for Computational Linguistics, New Orleans, LA, USA, 2018b, Online: http://aclweb.org/anthology/S18-2007.

Daniil Sorokin and Iryna Gurevych: 'Modeling Semantics with Gated Graph Neural Networks for Knowledge Base Question Answering', in: *Proceedings of COLING 2018, the 27th International Conference on Computational Linguistics (COLING)*, pp. 3306–3317, Association for Computational Linguistics, August 2018a, Online: http://aclweb.org/anthology/S18-2007.

Tommaso Soru, Edgard Marx, Diego Moussallem, Gustavo Publio, André Valdestilhas, Diego Esteves, and Ciro Baron Neto: 'SPARQL as a Foreign Language', in: *Proceedings of the 13th International Conference on Semantic Systems*, 2017.

Sasha Spala, Nicholas Miller, Franck Dernoncourt, and Carl Dockhorn: 'SemEval-2020 Task 6: Definition Extraction from Free Text with the DEFT Corpus', in: *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pp. 336–345, International Committee for Computational Linguistics, Barcelona (online), December 2020, Online: https://www.aclweb.org/anthology/2020.semeval-1.41.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov: 'Dropout: A Simple Way to Prevent Neural Networks from Overfitting', *Journal of Machine Learning Research* 15: 1929–1958, 2014.

Gabriel Stanovsky, Jessica Ficler, Ido Dagan, and Yoav Goldberg: 'Getting More Out Of Syntax with PropS', 2016, Online: http://arxiv.org/abs/1603.01648.

Jannik Strötgen and Michael Gertz: 'Multilingual and cross-domain temporal tagging', *Language Resources and Evaluation* 47 (2): 269–298, 2013.

Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum: 'Fast and Accurate Entity Recognition with Iterated Dilated Convolutions', in: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2670–2680, Copenhagen, Denmark, 2017.

Yu Su, Huan Sun, Brian Sadler, Mudhakar Srivatsa, Izzeddin Gur, Zenghui Yan, and Xifeng Yan: 'On Generating Characteristic-rich Question Sets for QA Evaluation', in: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 562–572, Austin, Texas, 2016.

Haitian Sun, Tania Bedrax-Weiss, and William Cohen: 'PullNet: Open Domain Question Answering with Iterative Retrieval on Knowledge Bases and Text', 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)*, pp. 2380–2390, Association for Computational Linguistics, Hong Kong, China, November 2019, Online: `https://www.aclweb.org/anthology/D19-1242`.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen: 'Open Domain Question Answering Using Early Fusion of Knowledge Bases and Text', in: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4231–4242, Association for Computational Linguistics, 2018, Online: `http://aclweb.org/anthology/D18-1455`.

Yaming Sun, Lei Lin, Duyu Tang, Nan Yang, Zhenzhou Ji, and Xiaolong Wang: 'Modeling mention, context and entity with neural networks for entity disambiguation', in: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1333–1339, Buenos Aires, Argentina, 2015.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning: 'Multi-instance Multi-label Learning for Relation Extraction', in: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 455–465, Jeju, Korea, 2012.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le: 'Sequence to Sequence Learning with Neural Networks', in Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger (Eds.): *Advances in Neural Information Processing Systems*, Vol. 27, pp. 3104–3112, Curran Associates, Inc., 2014.

Thomas Pellissier Tanon, Denny Vrandecic, Sebastian Schaffert, Thomas Steiner, and Lydia Pintscher: 'From Freebase to Wikidata: The Great Migration', in: *Proceedings of the World Wide Web Conference*, pp. 1419–1428, Montreal, Canada, 2016.

James Thorne and Andreas Vlachos: 'Automated Fact Checking: Task Formulations, Methods and Future Directions', in: *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*, pp. 3346–3359, Association for Computational Linguistics, Santa Fe, New Mexico, USA, August 2018, Online: https://www.aclweb.org/anthology/C18-1283.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal: 'FEVER: a Large-scale Dataset for Fact Extraction and VERification', in: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 809–819, Association for Computational Linguistics, New Orleans, Louisiana, June 2018, Online: https://www.aclweb.org/anthology/N18-1074.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal: 'Evaluating adversarial attacks against multiple fact verification systems', 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)*, pp. 2944–2953, Association for Computational Linguistics, Hong Kong, China, November 2019, Online: https://www.aclweb.org/anthology/D19-1292.

Ivan Titov and Alexandre Klementiev: 'A Bayesian Model for Unsupervised Semantic Parsing', in: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 1445–1455, Association for Computational Linguistics, Portland, Oregon, USA, June 2011, Online: https://www.aclweb.org/anthology/P11-1145.

Bayu Distiawan Trisedya, Gerhard Weikum, Jianzhong Qi, and Rui Zhang: 'Neural Relation Extraction for Knowledge Base Enrichment', in: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 229–240, Association for Computational Linguistics, Florence, Italy, July 2019, Online: https://www.aclweb.org/anthology/P19-1023.

Christina Unger, Axel-Cyrille Ngonga Ngomo, and Elena Cabrio: '6th Open Challenge on Question Answering over Linked Data (QALD-6)', in Harald Sack, Stefan Dietze, Anna Tordai, and Christoph Lange (Eds.): *Semantic Web Challenges*, pp. 171–177, Springer International Publishing, Cham, 2016.

Svitlana Vakulenko, Javier David Fernandez Garcia, Axel Polleres, Maarten de Rijke, and Michael Cochez: 'Message passing for complex question answering over knowledge graphs', in: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 1431–1440, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin: 'Attention is all you need', in: *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey

Hinton: 'Grammar as a Foreign Language', in: *Advances in Neural Information Processing Systems (NIPS)*, pp. 2755–2763, December 2015.

Andreas Vlachos and Sebastian Riedel: 'Fact Checking: Task definition and dataset construction', in: *Proceedings of the ACL Workshop on Language Technologies and Computational Social Science*, pp. 18–22, Baltimore, Maryland, USA, 2014.

Denny Vrandečić and Markus Krötzsch: 'Wikidata: A Free Collaborative Knowledgebase', *Communications of the ACM* 57 (10): 78–85, 2014.

Difeng Wang, Wei Hu, Ermei Cao, and Weijian Sun: 'Global-to-Local Neural Networks for Document-Level Relation Extraction', in: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 3711–3721, Association for Computational Linguistics, Online, November 2020, Online: `https://www.aclweb.org/anthology/2020.emnlp-main.303`.

Wei Wang, Ming Yan, and Chen Wu: 'Multi-Granularity Hierarchical Attention Fusion Networks for Reading Comprehension and Question Answering', in: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1705–1714, Association for Computational Linguistics, 2018, Online: `http://aclweb.org/anthology/P18-1158`.

Dirk Weissenborn, Tomáš Kočiskỳ, and Chris Dyer: 'Dynamic Integration of Background Knowledge in Neural NLU Systems', *arXiv preprint arXiv:1706.02596* 2018a.

Dirk Weissenborn, Pasquale Minervini, Isabelle Augenstein, Johannes Welbl, Tim Rocktäschel, Matko Bosnjak, Jeff Mitchell, Thomas Demeester, Tim Dettmers, Pontus Stenetorp, and Sebastian Riedel: 'Jack the Reader – A Machine Reading Framework', in: *Proceedings of ACL 2018, System Demonstrations*, pp. 25–30, Association for Computational Linguistics, Melbourne, Australia, July 2018b, Online: `https://www.aclweb.org/anthology/P18-4005`.

Peiyun Wu and Xiaowang Zhang: 'Improving Knowledge Base Question Answering with Question Understanding Augment', in Kerry L. Taylor, Rafael S. Gonçalves, Freddy Lécué, and Jun Yan (Eds.): *Proceedings of the ISWC 2020 Demos and Industry Tracks: From Novel Ideas to Industrial Practice co-located with 19th International Semantic Web Conference (ISWC)*, CEUR Workshop Proceedings Vol. 2721, pp. 167–171, CEUR-WS.org, 2020.

Shijie Wu and Mark Dredze: 'Beto, Bentz, Becas: The Surprising Cross-Lingual Effectiveness of BERT', 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)*, pp. 833–844, Association for Computational Linguistics, Hong Kong, China, November 2019, Online: `https://www.aclweb.org/anthology/D19-1077`.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao: 'Ques-

tion Answering on Freebase via Relation Extraction and Textual Evidence', in: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 2326–2336, Berlin, Germany, 2016.

Yi Yang and Ming-Wei Chang: 'S-MART: Novel Tree-based Structured Learning Algorithms Applied to Tweet Entity Linking', in: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pp. 504–513, Beijing, China, 2015.

Yilin Yang, Liang Huang, and Mingbo Ma: 'Breaking the Beam Search Curse: A Study of (Re-)Scoring Methods and Stopping Criteria for Neural Machine Translation', in: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 3054–3059, Association for Computational Linguistics, Brussels, Belgium, October-November 2018, Online: `https://www.aclweb.org/anthology/D18-1342`.

Xuchen Yao, Jonathan Berant, and Benjamin Van Durme: 'Freebase QA: Information Extraction or Semantic Parsing?', in: *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, pp. 82–86, Association for Computational Linguistics, Baltimore, MD, June 2014, Online: `https://www.aclweb.org/anthology/W14-2416`.

Yuan Yao, Deming Ye, Peng Li, Xu Han, Yankai Lin, Zhenghao Liu, Zhiyuan Liu, Lixin Huang, Jie Zhou, and Maosong Sun: 'DocRED: A Large-Scale Document-Level Relation Extraction Dataset', in: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 764–777, Association for Computational Linguistics, Florence, Italy, July 2019, Online: `https://www.aclweb.org/anthology/P19-1074`.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao: 'Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base', in: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pp. 1321–1331, Beijing, China, 2015.

Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh: 'The Value of Semantic Parse Labeling for Knowledge Base Question Answering', in: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 201–206, Berlin, Germany, 2016.

Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cicero dos Santos, Bing Xiang, and Bowen Zhou: 'Improved Neural Relation Detection for Knowledge Base Question Answering', in: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 571–581, Vancouver, Canada, 2017.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev: 'TypeSQL: Knowledge-Based Type-Aware Neural Text-to-SQL Generation', in: *Proceedings of the 2018 Conference of the North American Chapter of the Association*

for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers), pp. 588–594, Association for Computational Linguistics, 2018, Online: http://aclweb.org/anthology/N18-2093.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao: 'Distant Supervision for Relation Extraction via Piecewise Convolutional Neural Networks', in: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1753–1762, Lisbon, Portugal, 2015.

Michael Zhai, Johnny Tan, and Jinho D Choi: 'Intrinsic and extrinsic evaluations of word embeddings', in: *Proceedings of the 30th Conference of Association for the Advancement of Artificial Intelligence (AAAI)*, pp. 4282–4283, AAAI Press, 2016.

Ningyu Zhang, Shumin Deng, Zhanlin Sun, Guanying Wang, Xi Chen, Wei Zhang, and Huajun Chen: 'Long-tail Relation Extraction via Knowledge Graph Embeddings and Graph Convolution 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)*, pp. 3016–3025, Association for Computational Linguistics, Minneapolis, Minnesota, June 2019, Online: https://www.aclweb.org/anthology/N19-1306.

Ningyu Zhang, Shumin Deng, Zhanling Sun, Xi Chen, Wei Zhang, and Huajun Chen: 'Attention-Based Capsule Networks with Dynamic Routing for Relation Extraction', in: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 986–992, Association for Computational Linguistics, Brussels, Belgium, October-November 2018, Online: https://www.aclweb.org/anthology/D18-1120.

Han Zhao, Zhengdong Lu, and Pascal Poupart: 'Self-adaptive hierarchical sentence model', in: *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 4069–4076, Buenos Aires, Argentina, 2015.

Jie Zhou, Xu Han, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun: 'GEAR: Graph-based Evidence Aggregating and Reasoning for Fact Verification', in: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 892–901, Association for Computational Linguistics, Florence, Italy, July 2019, Online: https://www.aclweb.org/anthology/P19-1085.

Hao Zhu, Yankai Lin, Zhiyuan Liu, Jie Fu, Tat-Seng Chua, and Maosong Sun: 'Graph Neural Networks with Generated Parameters for Relation Extraction', in: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1331–1339, Association for Computational Linguistics, Florence, Italy, July 2019, Online: https://www.aclweb.org/anthology/P19-1128.

Publikationsverzeichnis des Verfassers

Teresa Botschen, Daniil Sorokin, and Iryna Gurevych: 'Frame- and Entity-Based Knowledge for Common-Sense Argumentative Reasoning', in: *Proceedings of the 5th Workshop on Argument Mining*, pp. 90–96, Association for Computational Linguistics, Brussels, Belgium, November 2018, Online: `https://www.aclweb.org/anthology/W18-5211`.

Michael Bugert, Yevgeniy Puzikov, Andreas Rücklé, Judith Eckle-Kohler, Teresa Martin, Eugenio Martínez-Cámara, Daniil Sorokin, Maxime Peyrard, and Iryna Gurevych: 'LSDSem 2017: Exploring Data Generation Methods for the Story Cloze Test', in: *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pp. 56–61, Association for Computational Linguistics, Valencia, Spain, April 2017, Online: `https://www.aclweb.org/anthology/W17-0908`.

Andreas Hanselowski, Hao Zhang, Zile Li, Daniil Sorokin, Benjamin Schiller, Claudia Schulz, and Iryna Gurevych: 'UKP-Athene: Multi-Sentence Textual Entailment for Claim Verification', in: *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pp. 103–108, Association for Computational Linguistics, Brussels, Belgium, November 2018, Online: `https://www.aclweb.org/anthology/W18-5516`.

Daniil Sorokin and Iryna Gurevych: 'Context-Aware Representations for Knowledge Base Relation Extraction', in: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1784–1789, Association for Computational Linguistics, Copenhagen, Denmark, September 2017a, Online: `https://www.aclweb.org/anthology/D17-1188`.

Daniil Sorokin and Iryna Gurevych: 'End-to-end representation learning for question answering with weak supervision', in Mauro Dragoni, Monika Solanki, and Eva Blomqvist (Eds.): *Semantic Web Challenges — 4th SemWebEval Challenge at ESWC 2017, Portoroz, Slovenia, May 28 – June 1, 2017, Revised Selected Papers*, Communications in Computer and Information Science Vol. 769, pp. 70–83, Springer, 2017b.

Daniil Sorokin and Iryna Gurevych: 'Interactive Instance-based Evaluation of Knowledge Base Question Answering', in: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 114–119, Association for Computational Linguistics, Brussels, Belgium, November 2018c, Online: `https://www.aclweb.org/anthology/D18-2020`.

Daniil Sorokin and Iryna Gurevych: 'Mixing Context Granularities for Improved Entity Linking on Question Answering Data across Entity Categories', in: *Procee-*

*dings of the Seventh Joint Conference on Lexical and Computational Semantics (*SEM)*, pp. 65–75, Association for Computational Linguistics, New Orleans, LA, USA, 2018a, Online: http://aclweb.org/anthology/S18-2007.

Daniil Sorokin and Iryna Gurevych: 'Modeling Semantics with Gated Graph Neural Networks for Knowledge Base Question Answering', in: *Proceedings of COLING 2018, the 27th International Conference on Computational Linguistics (COLING)*, pp. 3306–3317, Association for Computational Linguistics, August 2018b, Online: http://aclweb.org/anthology/S18-2007.