Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

FAKULTÄT
FÜR MATHEMATIK, INFORMATIK
UND NATURWISSENSCHAFTEN

DISSERTATION

Knowledge Graph Question Answering with Generative Language Models

Debayan Banerjee

Language Technology
Department of Informatics
Faculty of Mathematics, Informatics and Natural Sciences

Universität Hamburg
Hamburg, Germany

A cumulative thesis submitted for the degree of

Doctor rerum naturalium (Dr. rer. nat.)

2024

Knowledge Graph Question Answering with Generative Language Models

Dissertation submitted by: Debayan Banerjee

Date of Disputation: 29.08.2024

Supervisors:
Prof. Dr. Chris Biemann, Universität Hamburg
Prof. Dr. Ricardo Usbeck, Universität Hamburg and Leuphana Universität Lüneburg

Universität Hamburg, Hamburg, Germany
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics

Language Technology

Affidavit

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbst verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

I hereby declare upon oath that I have written the present thesis independently and have not used further resources and aids than those stated in the dissertation.

June 20, 2024

Date

Signature

(Debayan Banerjee)

To my mother
Arundhati Banerjee

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisors, Chris and Ricardo. I thank Chris for giving me this opportunity, for believing in me and constantly supporting me in my journey of research and thesis writing. I thank Ricardo for providing me with scientific and emotional support in the process of writing this thesis, and also showing me how to walk the longer path of a future in academia. Without my supervisors by my side, the completion of this thesis would not be possible.

I would like to thank my school and university teachers from India, who instilled in me a sense of pursuit for knowledge. I would like to acknowledge the role of Prof. Dr. Jens Lehmann, whose generous reply to an email of mine in 2017, brought me from India to Bonn. There, I pursued my masters In Computer Science, and along with my friend Mohnish Dubey, I worked on several important topics. I would like to thank the good people of Germany, who not only provided me with world-class and tuition-free education, but also the right atmosphere for raising a family. I am thankful to the University of Hamburg for providing me with the latest computing infrastructure and comfortable working conditions.

I would like to thank my friends in Bonn, Debanjan, Shabnam, Durai, Aniruddha, Saptarshi, Arindam, who provided much needed emotional and mental support. I am thankful to my German neighbours in Hamburg, who made us feel welcome and safe in a new city. Similarly, I am fortunate to have Indian friends in Hamburg, Sayan, Vivek, Kunal, Sarthak, Senthil, who made me feel at home in the city.

I value my colleagues at the Language Technology group, who supported me with stimulating conversation on research topics, and also life in Germany. In particular, I enjoyed collaborating with Özge and Seid on research projects. I would like to thank Timo Baumann, who was a post-doc with the group, for his role in selecting me for the PhD program. I also cherish the time spent talking with fellow countryman Abhik Jana.

I would like to thank the WISTS and ITMC groups at the University of Hamburg, with whom, I collaborated on research topics, which are not a part of this thesis, but were a source of great joy to me. In particular, I would like to thank Prof. Dr. Tilo Böhmann, Dr. Mathis Poser, Tom Lewandowski and Dr. Christina Wiethof.

I would like to thank my co-authors Pranav, Jivat and Arefa, who are brilliant bachelors students from India. They made important contributions and also made research more enjoyable for me. I would like to thank Sushil, who completed his masters from University of Hamburg, for doing stellar work on the DBLP_QuAD dataset.

I would like to thank the students of the SEMS group, led by Ricardo, for providing an intellectually stimulating environment close to my topics of interest. In particular, I enjoyed my conversations with Cedric, Junbo, Longquan and Tilahun. I hope to continue collaborating with them in the future.

I would like to thank my parents, Nitai Prasad Bandyopadhyay and Arundhati Banerjee, for raising me, nurturing me and loving me. I wish my mother were alive

to see me reach this stage. I would like to thank my brother Sayan Banerjee, whose brilliance and mental strength is a constant source of inspiration for me.

In the end, I would like to thank my wife Pooja Agarwal, who has been the anchor to my ship which has seen several storms in the sea of life. Without her immense sacrifice and silent dedication to raising our daughter Urmika, I would be lost at sea.

Use of Third-Party Software

I made use of Grammarly, which is an AI-based tool for correcting spelling and grammar, throughout the draft of this thesis. I thank the Leuphana University of Lüneburg for providing me access to the premium version of Grammarly. For the initial drafts in LaTeX, I used overleaf.com, and later, shifted to the ShareLaTeX instance provided by the Gesellschaft für Wissenschaftliche Datenverarbeitung mbH Göttingen. The figures used in this thesis were created on Google Drawings. The LaTeXtemplate for this thesis was taken from https://github.com/uhh-lt/thesis-template-uhh-lt-latex and modified. The German version of the abstract was translated using Google Translate from the English abstract, which was later corrected by my German-speaking colleagues.

The future depends on some graduate student
who is deeply suspicious of everything I have said.

— Geoffrey Hinton

Abstract

A Knowledge Graph (KG) is a data structure that stores information about the world in the form of nodes and edges. The nodes represent people, places, things etc., while the edges store the relationships between the nodes. The nodes are also known as *entities*, while the edges are known as *relations* or *predicates*. Several popular search engines today make use of such KGs in the background. Some well-known and freely available KGs are DBpedia and Wikidata.

One way to access information from a KG is through Question Answering. For example, web-based search engines today give people the ability to type their questions and receive answers. Unfortunately, the current state of search engines leaves much to be desired in the complexity of the questions that a user may type. Current search engines work best when the search term is a keyword or a set of words. Processing complete sentences, with complex logical rules, is still an open problem.

One large step in the direction of language understanding has been the arrival of pre-trained Language Models, such as BERT. Such models have been trained on large amounts of text corpus, and surprisingly, some variants of these models, such as T5 and BART, develop a remarkable ability to generate text, the likes of which are difficult to distinguish from that produced by a human author. These models are also called generative Language Models and are a central focus of this thesis.

Given a question by a user, how does one fetch an answer from the KG? This task is commonly known as Knowledge Graph Question Answering (KGQA). One of the techniques is to convert the user's question to a logical form, or a structured query. One popular query language the reader might be familiar with is SQL. SQL, though, is appropriate for relational databases. In the KG world, the analogue would be a language called SPARQL. The task of converting the natural language text to a logical form is known as *semantic parsing*.

To be able to execute a SPARQL query on a KG, the SPARQL schema must be valid, e.g., it must be syntactically correct, and it should be logically correct, e.g., one can not expect the correct answer if AND is replaced with an OR. The other requirement is that the constants in the query, such as entity and relation IDs, have to be placed in the correct manner in the query. *In this thesis, we explore the abilities of generative Language Models (LMs) in the task of KGQA,* with a focus on the semantic parsing approach.

This dissertation hypothesizes that generative LMs can be used effectively for the task of KGQA. We form two research questions over this hypothesis, and to answer these research questions, a series of five topically interconnected publications are presented in a cumulative fashion. We first test two popular generative LMs on the task of semantic parsing. We compare the performance of these models to their non-pre-trained predecessors. To this end, we utilize some well-known and openly available datasets, which address well-known KGs. Later, we develop methods to improve the default performance of such models on this task.

This thesis shows that while generative LMs are not ideal for the semantic parsing task in their default mode, special text-handling mechanisms can be incorporated into the model to improve their performance considerably. With these adaptations, the models produce state-of-the-art performance across five datasets that address four different KGs.

This thesis provides researchers in this field with a set of tools and techniques to work with generative LMs and adapt them appropriately to the task of KGQA. We show in subsequent chapters, that our findings have been used by contemporary researchers to further advance the state-of-the-art. In the end, we produce a new KGQA dataset, built over a smaller domain-specific KG. On this dataset, a challenge was organized in which seven participating teams tried the latest methods, and further pushed the state-of-the-art for this task. All the code and data used and developed in this thesis have been released as open-source.

Zusammenfassung

Ein Knowledge Graph (KG) ist eine Datenstruktur, die Informationen über die Welt in Form von Knoten und Kanten speichert. Die Knoten stellen Personen, Orte, Dinge usw. dar, während die Kanten die Beziehungen zwischen den Knoten speichern. Die Knoten werden auch als *entities* bezeichnet, während die Kanten als *relations* oder *predicates* bekannt sind. Viele gängige Suchmaschinen machen sich heute solche KGs im Hintergrund zunutze. Einige bekannte und frei verfügbare KGs sind DBpedia und Wikidata.

Eine Möglichkeit, auf Informationen eines KG zuzugreifen, ist die Beantwortung von Fragen. Beispielsweise bieten webbasierte Suchmaschinen heute den Menschen die Möglichkeit, ihre Fragen einzugeben und Antworten zu erhalten. Leider lässt die aktuelle Performance von Suchmaschinen hinsichtlich der Komplexität der Fragen, die ein Benutzer eingeben kann, zu wünschen übrig. Aktuelle Suchmaschinen funktionieren am besten, wenn der Suchbegriff ein Schlüsselwort oder eine Reihe von Wörtern ist. Die Verarbeitung vollständiger Sätze mit komplexen logischen Regeln ist immer noch ein offenes Problem.

Ein großer Schritt in Richtung Sprachverständnis war die Einführung vortrainierter Sprachmodelle wie BERT. Solche Modelle wurden an großen Textkorpus trainiert, und überraschenderweise entwickeln einige Varianten dieser Modelle, wie etwa T5 und BART, eine bemerkenswerte Fähigkeit, Texte zu generieren, die sich nur schwer von Texten unterscheiden lassen, die von einem menschlichen Autor erstellt wurden . Diese Modelle werden auch generative Sprachmodelle genannt und stehen im Mittelpunkt dieser Arbeit.

Wie erhält man auf eine Frage eines Benutzers eine Antwort von einem KG? Diese Aufgabe wird allgemein als Knowledge Graph Question Answering (KGQA) bezeichnet. Eine der Techniken besteht darin, die Frage des Benutzers in eine logische Form oder eine strukturierte Abfrage umzuwandeln. Eine beliebte Abfragesprache, mit der der Leser möglicherweise vertraut ist, ist SQL. SQL eignet sich für relationale Datenbanken. In der KG-Welt wäre das Analogon eine Sprache namens SPARQL. Die Aufgabe, den Text in natürlicher Sprache in eine logische Form umzuwandeln, wird als *semantisches Parsen* bezeichnet.

Um eine SPARQL-Abfrage auf einem KG ausführen zu können, muss das SPARQL-Schema gültig sein, d. h. es muss syntaktisch korrekt sein und es sollte logisch korrekt sein, d. h. man kann nicht die richtige Antwort erwarten, wenn AND durch ein OR ersetzt wird . Die andere Anforderung besteht darin, dass die Konstanten in der Abfrage, wie z. B. *Entity* und *Relation*-IDs, in der richtigen Weise in der Abfrage platziert werden müssen. *In dieser Arbeit untersuchen wir die Fähigkeiten generativer Sprachmodelle (LMs) in KGQA*, wobei der Schwerpunkt auf dem semantischen Parsing-Ansatz liegt.

In dieser Dissertation wird die Hypothese aufgestellt, dass generative LMs effektiv für KGQA eingesetzt werden können. Wir formulieren zwei Forschungsfragen zu dieser Hypothese, und um diese Forschungsfragen zu beantworten, wird eine Reihe von fünf thematisch miteinander verbundenen Veröffentlichungen kumulativ vorgestellt. Wir

testen zunächst zwei beliebte generative LMs hinsichtlich der Aufgabe des semantischen Parsens. Wir vergleichen die Leistung dieser Modelle mit ihren nicht vorab trainierten Vorgängern. Hierzu nutzen wir einige bekannte und frei verfügbare Datensätze, die sich an bekannte KGs richten. Später entwickeln wir Methoden, um die Standardleistung solcher Modelle bei dieser Aufgabe zu verbessern.

Diese Arbeit zeigt, dass generative LMs zwar in ihrem Standardmodus nicht ideal für die semantische Parsing-Aufgabe sind, dass jedoch spezielle Textverarbeitungsmechanismen in das Modell integriert werden können, um ihre Leistung erheblich zu verbessern. Mit diesen Anpassungen liefern die Modelle eine Leistung auf Höhe der Zeit auf fünf Datensätzen, die vier verschiedene KGs ansprechen.

Diese Arbeit stellt Forschern auf diesem Gebiet eine Reihe von Werkzeugen und Techniken zur Verfügung, um mit generativen LMs zu arbeiten und sie entsprechend an die Aufgabe der KGQA anzupassen. In den folgenden Kapiteln zeigen wir, dass unsere Erkenntnisse von Forschern genutzt wurden, um den Stand der Technik weiter voranzutreiben. Am Ende erstellen wir einen neuen KGQA-Datensatz, der auf einem kleineren domänenspezifischen KG aufbaut. Anhand dieses Datensatzes wurde eine *Challenge* organisiert, bei der sieben teilnehmende Teams die neuesten Methoden ausprobierten und den Stand der Technik für diese Aufgabe weiter vorantrieben. Der gesamte in dieser Arbeit verwendete und entwickelte Code und die Daten wurden als Open Source veröffentlicht.

Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
GPT	Generative Pre-trained Transformer
KG	Knowledge Graph
KGQA	Knowledge Graph Question Answering
LM	Language Model
LLM	Large Language Model
ML	Machine Learning
NLP	Natural Language Processing
PLM	Pre-Trained Language Model
T2TPLM	Text-to-Text Pre-Trained Language Model

Note: The terms *generative LM* and *T2TPLM* are used interchangeably throughout this thesis. When referring to generative LMs or T2TPLMs, we consider specifically Transformer-based pre-trained models that take text as input and produce text as output. We refrain from using the term GPT for generic generative LMs, as GPT is also the name of a particular generative LM released by the private enterprise OpenAI.

1 Introduction

1.1 Motivation and Research Objectives

A Knowledge Graph (KG) is an information store that represents knowledge in the form of node-edge-node triples. The nodes are known as entities, while the edges are called properties or relations. As seen in Figure 1.1, in a general-purpose KG, the entities are usually persons, places, and objects of interest, while the properties define the relationships between the entities. KGs are typically human-curated and are guaranteed to contain correct information. Wikidata (Vrandečić and Krötzsch, 2014), DBpedia (Lehmann, Isele, et al., 2015), Freebase (Bollacker et al., 2008), and Yago (Suchanek et al., 2007) are examples of well-known general-purpose KGs.

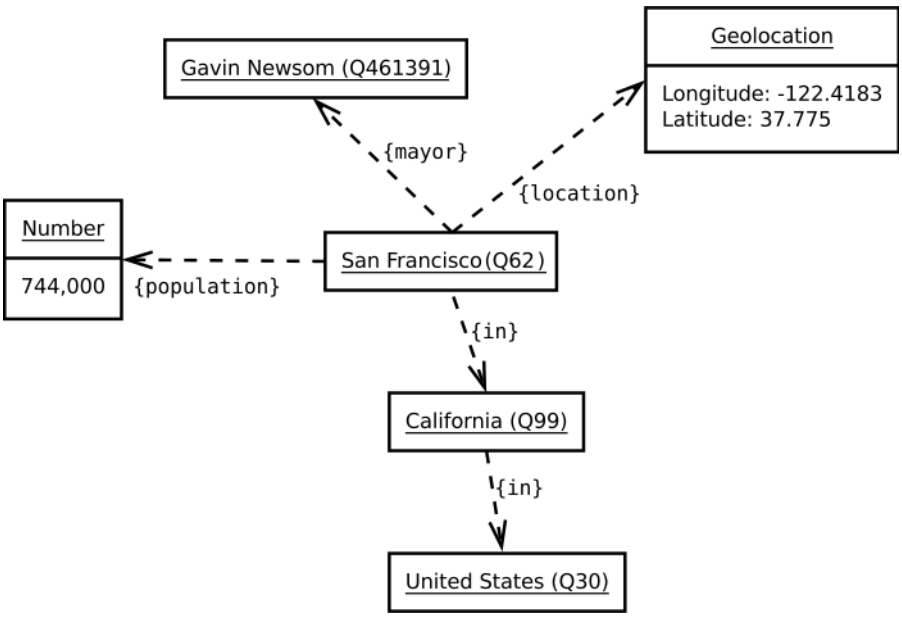


Figure 1.1: A sub-graph sample from Wikidata KG.
Source: Wikidata (2012)

Apart from general-purpose knowledge graphs, domain-specific knowledge graphs also exist. For example, knowledge graphs belonging to medical (Cui et al., 2023), financial (S Wang, 2022), scholarly (Färber et al., 2023) or music (Rashid et al., 2018) domains. More recently, the concept of personal knowledge graphs (Balog and Kenter, 2019) have emerged, which may store details of our everyday activities on a personal server. For a detailed description of the history of the development of Knowledge Graphs, the reader may refer to Gutierrez and Sequeda (2021).

The objective of Knowledge Graph Question Answering (KGQA) (Höffner et al., 2017; Lan, G He, et al., 2021), is to fetch answers from a KG, given a question in natural language as input, e.g., `What are the coordinates of the city for which Gavin Newsom was a mayor?`. (See Figure 1.1).

Typically, the first step in KGQA is to identify spans of entities in the given text. This task is also called Named Entity Recognition (NER). Entities are distinct and identifiable objects or concepts, often nouns, such as people, places, and things. In the sample question above, the entity would be `Gavin Newsom`.

The next step is to find the corresponding node for this given entity in the KG. This task may be challenging, because multiple people may share the same name, and finding the right node would require further *disambiguation*. This step is called Entity Linking (EL). For an overview of the challenges for this task, one may refer to the surveys of Sevgili et al. (2022) and Möller et al. (2022).

Once the correct entity node has been identified in the KG, to successfully answer the question, the neighboring edges and nodes have to be further analyzed. For example, to answer the given question, two edges in the KG are of interest: namely `mayor` and `location`. Identifying the correct edges is a task called Relation Linking (RL).

At this stage, some systems fetch the answer based on further exploration of the neighborhood so discovered, especially in cases where the answer is also another node in the graph. This method falls into the category of the *retrieval-based approach*. However, for more complex or multi-hop questions, such as queries that require a count or aggregation (e.g.: `How many cities in the USA have a population above 500,000?`) the answer does not lie in a node or edge in the graph, but instead, a formal query must be generated to compute the total. Moreover, the generation of logical forms has the added benefit of *explainability*. That is, one can read and understand the generated logical form to understand the final answer and how it was retrieved. The task of generating a formal query is also known as *semantic parsing*. In In this thesis, we place greater emphasis on the semantic parsing approach, i.e., we focus on the model's ability to generate logical forms.

In recent times, the Transformer architecture (Vaswani et al., 2017) has shown remarkable performance in a variety of Natural Language Processing (NLP) tasks, including Question Answering (QA). The Transformer architecture's parallel encoder design allows efficient training over a large corpus of text, and this feature has led to the arrival of the pre-trained Transformer-based architecture.

GPT-1 Radford et al. (2018) was the first pre-trained generative Language Model (LM) architecture, developed by the enterprise OpenAI and released in 2018. GPT-1 showed the capabilities of pre-trained models on the generative tasks, however, it was not freely available to the broad audience for download. OpenAI decided to restrict its usage to control misuse of its abilities. It was finally early 2020 when Google and Facebook almost simultaneously released T5 (Raffel et al., 2020) and BART (M Lewis et al., 2020), generative models everyone could download, use, fine-tune, and adapt to individual use

cases. The models came in versions that were around 200M parameters in size, which fit neatly into most GPUs (abbreviation for graphical processing units, which are hardware required for deep learning) in an academic setting. The reader may find additional details regarding these two families of models in Subsections 2.6.3 and 2.6.3 of this thesis.

The research work for this thesis also began in 2020, and as a natural consequence of the release of freely usable generative LMs, we decided to try them on the KGQA task. Based on existing literature at the time, we found no previous work on pre-trained Transformer-based generative Language Models on the KGQA task.

Part of the justification for applying generative models to the KGQA task, is the first half of any QA task, which is language understanding. Beyond any doubt, pre-trained LMs set new benchmarks in a majority of Natural Language Understanding (NLU) tasks[1]. However, the second part of KGQA, which is the formation of the logical form, or fetching of the answer from a KG, does not come naturally to generative LMs. Generative models have no natural connection to an external knowledge store, or an inbuilt mechanism to perform information retrieval. Moreover, the corpus on which T5 and BART were pre-trained, contained no examples of question-to-query mappings, although there were larger examples of description-to-code samples from GitHub. Hence, it could not be expected that these models perform well out of the box on our task of *semantic parsing* without additional fine-tuning.

It is the combined effect of the arrival of generative LM technology, and the scope of open research questions on the suitability of these models on the KGQA tasks, which motivated the research direction of this thesis.

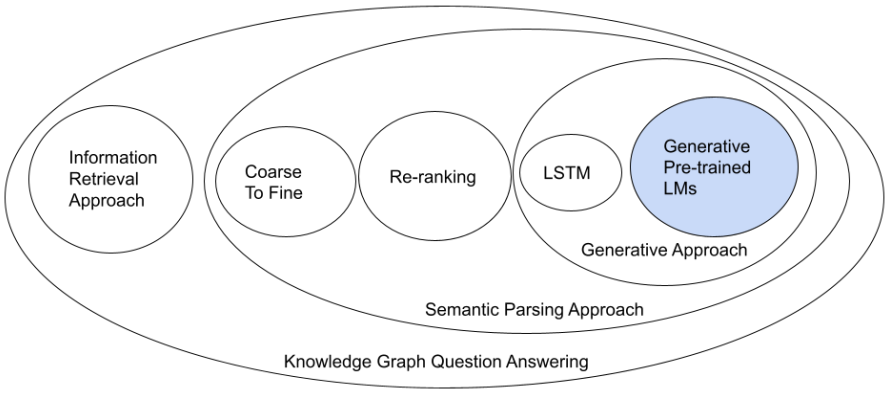


Figure 1.2: Situating the thesis topic in the KGQA landscape.

1.2 Related Work

Early work on KGQA focused on answering simple questions (Lopez, Motta, and Uren, 2006; Lopez and Motta, 2004), where a single triple of fact is involved, e.g., `Where was Mahatma Gandhi born?` can be answered using the triple `<Mahatma Gandhi, birthplace, Porbandar India>`. At the time, finding the right mapping between a sentence and a triple often relied on finding synsets from Wordnet (Miller, 1995) and finding lexical matches. As the field of machine learning progressed further, modern KGQA systems started using neural networks, e.g., Lukovnikov, Fischer, et al. (2017).

[1] https://gluebenchmark.com/

In the case of simple questions, the mere identification of the correct triple is sufficient. However, for more complex or multi-hop questions, such as: `How many rivers flow through the most populous nation?`, the answer does not lie in the node in a graph, and instead, a formal query has to be formed. Early works on semantic parsing (LS Zettlemoyer and Collins, 2005; L Zettlemoyer and Collins, 2007; Berant, Chou, et al., 2013a; Reddy et al., 2014; Artzi and L Zettlemoyer, 2013; Unger et al., 2012) made use of Combinatorial Categorical Grammars (Bar-Hillel, 1953) (CCG), or rule-based parsing to generate logical forms such as SPARQL or $\lambda - DCS$. Unfortunately, designing CCG rules can be complex and labor-intensive. Crafting rules for all possible sentence structures in a language requires a deep understanding of linguistics and extensive rule engineering. For further information on semantic parsing techniques in the pre-neural network era, we refer the reader to the surveys of Kamath and Das (2018), Kumar and Bedathur (2020), and Q Zhu, X Ma, et al. (2019).

In their survey on KGQA semantic parsing, Gu, Kase, et al. (2021) outline three main approaches widely adopted in neural semantic parsing for KGQA: *ranking methods, coarse-to-fine methods, and generative methods*. Ranking methods (Abujabal et al., 2017; Berant and Liang, 2014; Ye et al., 2022; Zafar et al., 2018) initially generate a list of candidate paths, spanning entities and relations from the KG and rely on computing the matching score for each candidate-path and question pair. Coarse-to-fine methods (Bhutani et al., 2019; Das et al., 2021; Y Sun et al., 2020; Zhang et al., 2019) first create query skeletons and then connect these skeletons to the KG using permissible schema items. The discovery of permissible schema may be made by exploring the KG nodes and their neighbourhoods, or in cases where an ontology is available, following the rules specified in the ontology. For example, an invalid schema item would be a case where we try to connect an edge of type `spouse` to a node of type `place`. Generative methods typically employ models such as LSTMs (Hochreiter and Schmidhuber, 1997) or GRUs (Cho et al., 2014) which can produce the query as output via step-by-step decoding.

Despite this categorization by Gu, Kase, et al. (2021), an end-to-end KGQA system may fall into multiple categories. For example, our work **GETT-QA: Graph Embedding based T2T Transformer for Knowledge Graph Question Answering** first generates a skeleton query which falls under the *coarse-to-fine* method. In a subsequent step, candidate entities and relations are re-ranked, which may fall in the *re-ranking approach*. Additionally, since the initial coarse query is generated by a generative LM, namely T5 (Raffel et al., 2020), this system also belongs to the *generative approach*.

We explain the generative approaches in more detail, as this correlates closely with the focus of our thesis: Early efforts in the generative approach have explored KGQA semantic parsing using encoder-decoder models, such as LSTMs (Hochreiter and Schmidhuber, 1997). Soru, Marx, Valdestilhas, et al. (2018), Diomedi and Hogan (2021), Luz and Finger (2018) utilize the machine translation approach using LSTM networks for translating questions to corresponding SPARQL queries. In order to be syntactically valid, logical forms must obey a set of grammatical rules. These rules can often be suitably represented as a tree. As a result, several works introduced *structured decoding* mechanisms so that the decoded logical form follows the tree-based grammar structure. For example, Dong and Lapata (2016) proposes a tree decoding model that decodes a query tree in a top-down breadth-first order. Alvarez-Melis and Jaakkola (2017) introduce an improved tree decoder , where the parent-to-child and sibling-to-sibling information flows are modelled through two different RNNs. Cheng, Reddy, et al. (2017), Cheng and Lapata (2018), and Cheng, Reddy, et al. (2019) propose a transition-based

neural semantic parser that adapts the Stack-LSTM proposed by Dyer et al. (2015). Zafar et al. (2018) exploit syntax to train a query ranking model leveraging the Tree-LSTM (Tai et al., 2015) model. Encoder-decoder-based approaches typically produce sequences in left-to-right order, which is also commonly known as auto-regressive decoding. This does not fall naturally into the tree-based structure of grammar. Moreover, for most query languages, such as SQL and SPARQL, the order of the logical clauses does not matter. Hence, two queries that produce the same result but vary in order of clauses will not be equally rewarded during the decoding step. As a result, non-autoregressive decoding strategies appeared (Lukovnikov and Fischer, 2021; Q Zhu, Khan, et al., 2020; Stern et al., 2019), which not only resulted in better grammar-styled productions, but also achieved sub-linear decoding time complexity.

In more recent times, after the arrival of pre-trained language models, non-generative pre-trained models such as BERT (Devlin et al., 2019), which is based on the Transformer architecture, have also been used for the task of KGQA semantic parsing (Zhang et al., 2019; Yin et al., 2021; Gu, Kase, et al., 2021). It is to be noted that since BERT is not a generative model, the use of BERT generally falls under ranking or coarse-to-fine approaches. The most common strategy is to generate question-query pairs and use BERT for scoring and ranking these pairs.

Coming back to the topic of generation methods for semantic parsing, in this thesis, we explore generative pre-trained language models such as T5 (Raffel et al., 2020) and BART (M Lewis et al., 2020) for the first time on the semantic parsing task. T5 and BART, like BERT, are also pre-trained language models. However, while BERT generates embeddings as output and only has an encoder, T5 and BART can generate text as output, like their predecessors, LSTMs, and contain both an encoder and a decoder. For more details on these models, we refer the reader to Section 2.6.3.

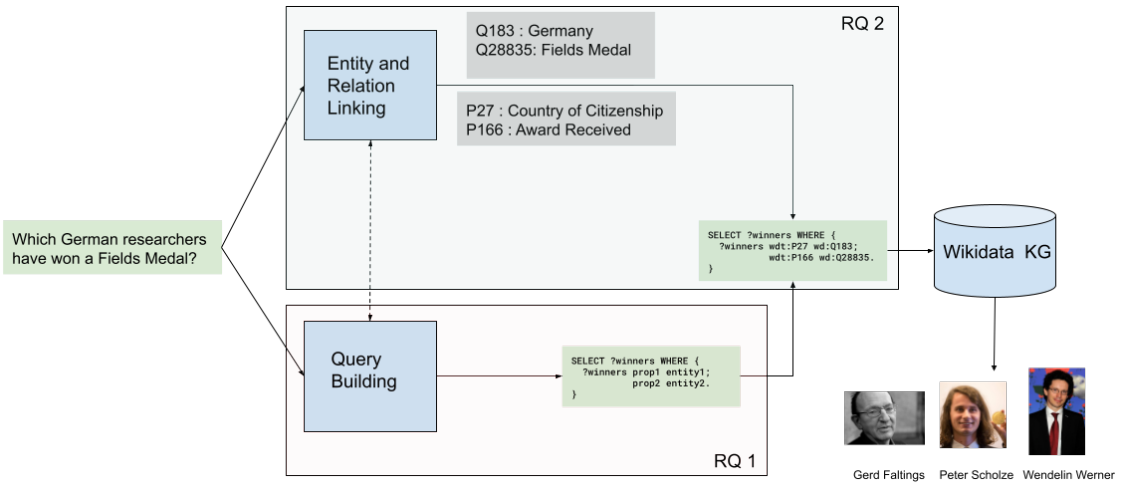


Figure 1.3: A typical semantic parsing based KGQA architecture, with our research questions situated in it.

1.3 Hypothesis and Research Questions

To set a background for our research questions, let us consider a motivating example. For the given question:

*Is the Poisson's ratio of gold equal to **0.4**?*

the correct logical form structure for SPARQL would be:

```
ASK WHERE {
        ENT1 REL1 ?obj
        filter ( ?obj = LIT1 )
}
```

The query above depicts a skeleton of the correct SPARQL query. In its current form, this query is not yet executable on a KG endpoint, and will not fetch the desired results. However, the generation of this skeleton query is a challenging task on its own and is the first step toward forming a complete query. For the query to be complete, the logical form has to be *grounded*, i.e., the placeholders ENT1, REL1, and LIT1 must be replaced with corresponding entity and relation IDs from the KG, and the appropriate literal must be extracted from the question and replaced in the skeleton query. Once this process is completed, the final query is as follows:

```
ASK WHERE {
    wd:Q897 wdt:P5593 ?obj
    filter(?obj = 0.4)
}
```

`wd:Q897`[2] is the entity ID for `Gold` while `wdt:P5593`[3] is the relation ID for `Poisson's ratio`. This time, the placeholders have been substituted with corresponding entity and relation IDs, and the relevant literal `0.4` has been extracted from the input question, and substituted in place of the literal placeholder LIT1. The query above, when executed on the Wikidata endpoint[4] produces the correct response.

With this context in mind, our singular research hypothesis is as follows:

Hypothesis

Generative Language Models can be used effectively for the task of Knowledge Graph Question Answering.

which gives rise to the following research questions:

Research Question 1

Can generative Language Models produce correct logical form structure?

Research Question 2

Can generative Language Models aid in grounding of logical forms to the Knowledge Graph?

[2] https://www.wikidata.org/wiki/Q897
[3] https://www.wikidata.org/wiki/Property:P5593
[4] https://query.wikidata.org/

1.4 Contributions

In this section, we briefly describe the contributions of this thesis vis-a-vis our research questions.

Contributions to RQ 1: Can generative LMs generate correct logical form structure?

Addressing RQ1, in our paper **Modern Baselines for SPARQL Semantic Parsing** (Banerjee, Nair, Kaur, et al., 2022), we test T5, BART and a Pointer Generator Network on the task of KGQA semantic parsing. The task of the models is to produce the correct SPARQL query, given the question, and the entities and relations as input. In this task, the entities and relations are provided beforehand, and no additional linking step is necessary. We discover that T5 works best, but only when special characters and symbols are appropriately handled and replaced with special tokens from the T5 tokenizer vocabulary.

Additionally, in our paper **The Role of Output Vocabulary in T2T LMs for SPARQL Semantic Parsing** (Banerjee, Nair, Usbeck, et al., 2023b), we discover that, when the SPARQL vocabulary is replaced with a linguistic vocabulary the semantic parsing accuracy of a generative LM improves because a generative LM tends to interface better with natural language than abstract programming vocabulary.

Contributions to RQ 2: Can generative LMs aid in the grounding of logical forms to the KG?

Addressing RQ2, in our paper **GETT-QA: Graph Embedding based T2T Transformer for Knowledge Graph Question Answering** (Banerjee, Nair, Usbeck, et al., 2023a), we develop a generative LM-based end-to-end KGQA system named GETT-QA, that generates a logical form, along with entity and relation labels. To aid in grounding these labels to the precise nodes and edges in a KG, the model is additionally trained to generate a shorter form of KG embeddings. These embeddings are used to disambiguate between entity candidates, leading to the construction of a full SPARQL query.

Additionally, in our paper **DBLPLink: An Entity Linker for the DBLP Scholarly Knowledge Graph** (Banerjee, Arefa, et al., 2023), we develop DBLPLink, which is a web-based demonstration of a generative LM-based entity linker, over the DBLP KG. The DBLP KG (Ley, 2002) is a computer science bibliography in the RDF[5] format. We show that generative LMs can act as strong entity label generators as the initial step of the larger goal of entity linking.

Finally, in our paper **DBLP-QuAD: A Question Answering Dataset over the DBLP Scholarly Knowledge Graph** (Banerjee, Awale, et al., 2023), we develop DBLP_QuAD, which is a KGQA dataset over the DBLP KG. It contains SPARQL annotations for corresponding questions. Most SPARQL annotated KGQA datasets address general KGs like Wikidata and DBpedia. With DBLP_QuAD, we introduced a domain-specific KGQA dataset that can

[5] https://www.w3.org/TR/2004/REC-rdf-concepts-20040210/

be used for KGQA semantic parsing research. This addresses both RQ1 and RQ2, since forming the correct SPARQL query requires the construction of the correct logical form, and also grounding of entity and relations.

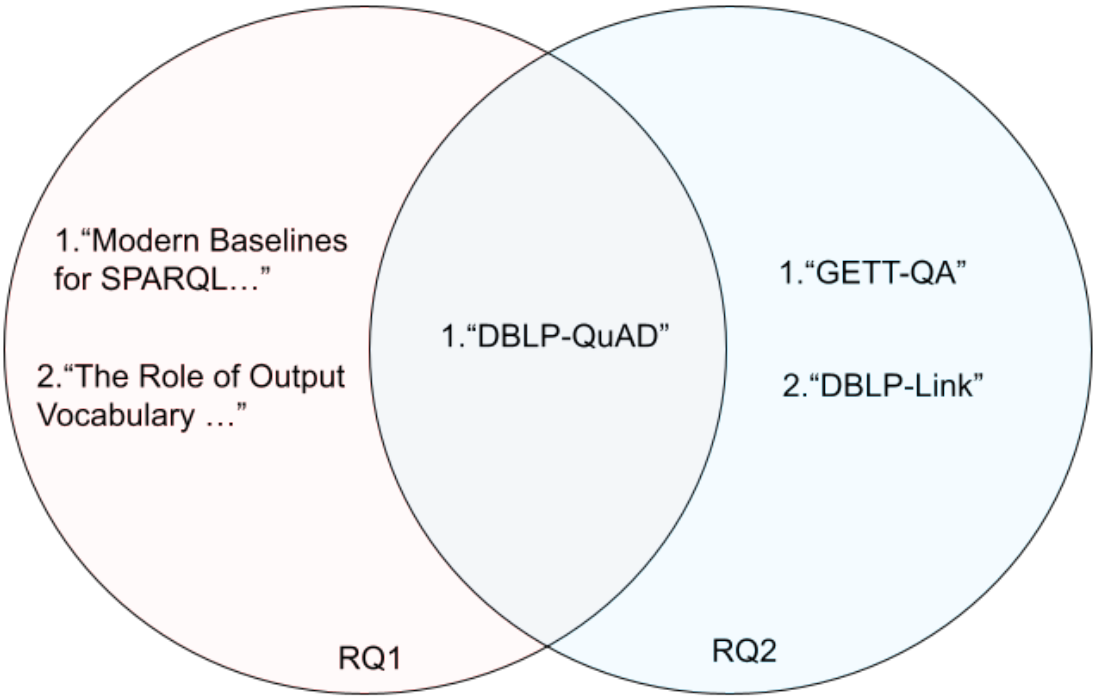


Figure 1.4: Situating our papers vis-à-vis our research questions visually, to be seen with reference to Section 1.5.

1.5 Publications

In this section, we provide a list of accepted papers that comprise this thesis. Additionally, the contributions of each author of the accepted papers are described in more detail. I am the first author of each of the papers. For the paper **The Role of Output Vocabulary in T2T LMs for SPARQL Semantic Parsing**, Pranav Ajit Nair and I are joint first authors with equal contributions.

1.5.1 Accepted Papers Comprising This Thesis

- **Banerjee, D.**, Nair, P. A., Kaur, J. N., Usbeck, R., Biemann, C. 2022. **Modern Baselines for SPARQL Semantic Parsing**, In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 2260–2265, Madrid, Spain. Association for Computing Machinery. Research Track, Short Paper. (Banerjee, Nair, Kaur, et al., 2022)

- **Banerjee, D.**, Awale, S., Usbeck R., Biemann, C. 2023. **DBLP-QuAD: A Question Answering Dataset over the DBLP Scholarly Knowledge Graph**, In Proceedings

of the 13th International Workshop on Bibliometric-enhanced Information Retrieval @ The 45th European Conference for Information Retrieval, pages 37-51, Dublin, Ireland. Full Paper. (Banerjee, Awale, et al., 2023)

- **Banerjee, D.**, Nair P. A., Usbeck R., Biemann, C. 2023. **GETT-QA: Graph Embedding-based T2T Transformer for Knowledge Graph Question Answering**, In Proceedings of the 20th Extended Semantic Web Conference, pages 279–297, Hersonissos, Greece. Research Track, Full Paper. (Banerjee, Nair, Usbeck, et al., 2023a)

- **Banerjee, D.**, Nair, P. A., Usbeck, R., Biemann, C. 2023. **The Role of Output Vocabulary in T2T LMs for SPARQL Semantic Parsing**. In Findings of the Association for Computational Linguistics: ACL 2023, pages 12219–12228, Toronto, Canada. Association for Computational Linguistics. Short Paper. (Banerjee, Nair, Usbeck, et al., 2023b)

- **Banerjee D.**, Arefa, Usbeck R., Biemann C. (2023): **DBLPLink: An Entity Linker for the DBLP Scholarly Knowledge Graph**, In Proceedings of the ISWC 2023 Posters, Demos and Industry Tracks co-located with 22nd International Semantic Web Conference, Athens, Greece. Demo paper. (Banerjee, Arefa, et al., 2023)

1.5.2 Comments on the degree of authorship

In our paper **Modern Baselines for SPARQL Semantic Parsing** (Banerjee, Nair, Kaur, et al., 2022), I conceptualized the experimental basis of the paper, using generative LMs for the task of KGQA semantic parsing. Jivat Neet Kaur performed the experiments with the Pointer Generator Network (PGN), while I performed the T5 experiments. I wrote the first draft of the paper, which Jivat helped improve with her suggestions, and also provided the PGN diagram used in the paper. Jivat left the project at this point, and Pranav Ajit Nair joined. Pranav performed a new set of experiments with T5, while I performed a new set of PGN experiments. Pranav helped improve the second draft of the paper with his suggestions.

In our paper **DBLP_QuAD: A Question Answering Dataset over the DBLP Scholarly Knowledge Graph** (Banerjee, Awale, et al., 2023), I conceptualized the DBLP_QuAD dataset. Sushil Awale improved the concept with additional templates and implemented the code and methods to generate the datasets. He performed the baseline experiments. Sushil and I collaborated in manually controlling the quality of the generated questions, and also in writing the paper.

In our paper **GETT-QA: Graph Embedding-based T2T Transformer for Knowledge Graph Question Answering** (Banerjee, Nair, Usbeck, et al., 2023a), I conceptualized the GETT-QA KGQA pipeline. I wrote the code, performed the experiments, and wrote the paper. Pranav Ajit Nair verified the correctness of the experiments and proofread the paper.

In **The Role of Output Vocabulary in T2T LMs for SPARQL Semantic Parsing** (Banerjee, Nair, Usbeck, et al., 2023b), I conceptualized the experimental basis of trying different output vocabularies with generative LMs for semantic parsing. Pranav Ajit Nair came up with the idea of trying prompt tuning apart from fine-tuning. He carried out the experiments, while I verified the correctness of the experiments. Pranav and I collaborated equally in writing the paper.

In **DBLPLink: An Entity Linker for the DBLP Scholarly Knowledge Graph** (Banerjee, Arefa, et al., 2023), I conceptualized the design of the DBLPLink entity linker. Arefa implemented the entity label generator using T5, while I implemented the entity-re-ranker for disambiguation. Arefa developed an initial version of the web demo, which I improved for performance reasons. I wrote the paper, which Arefa proofread.

Prof. Dr. Chris Biemann and Prof. Dr. Ricardo Usbeck provided overall supervisory guidance for all the papers that are a part of this thesis.

1.6 Adaptation Disclosure

Sections through 1.1 to 1.3 have been adapted from my PhD symposium paper presented at the Extended Semantic Web Conference 2023 (Banerjee, 2023).

1.7 Thesis Outline

As depicted in Figure 1.5, this thesis is divided into eight chapters. Chapters 1, 2 and 8 sit as a wrapper around the core research papers, and address the broad scope of the thesis, presenting a unified and over-arching discussion on the general aspects of our research. Chapters 3 to 7 are our peer-reviewed and published papers that address the research questions of our thesis. The papers presented therein are verbatim copies of the original respective proceedings, except that the bibliographies of all the papers have been combined and unified in style, and presented at the end of the thesis.

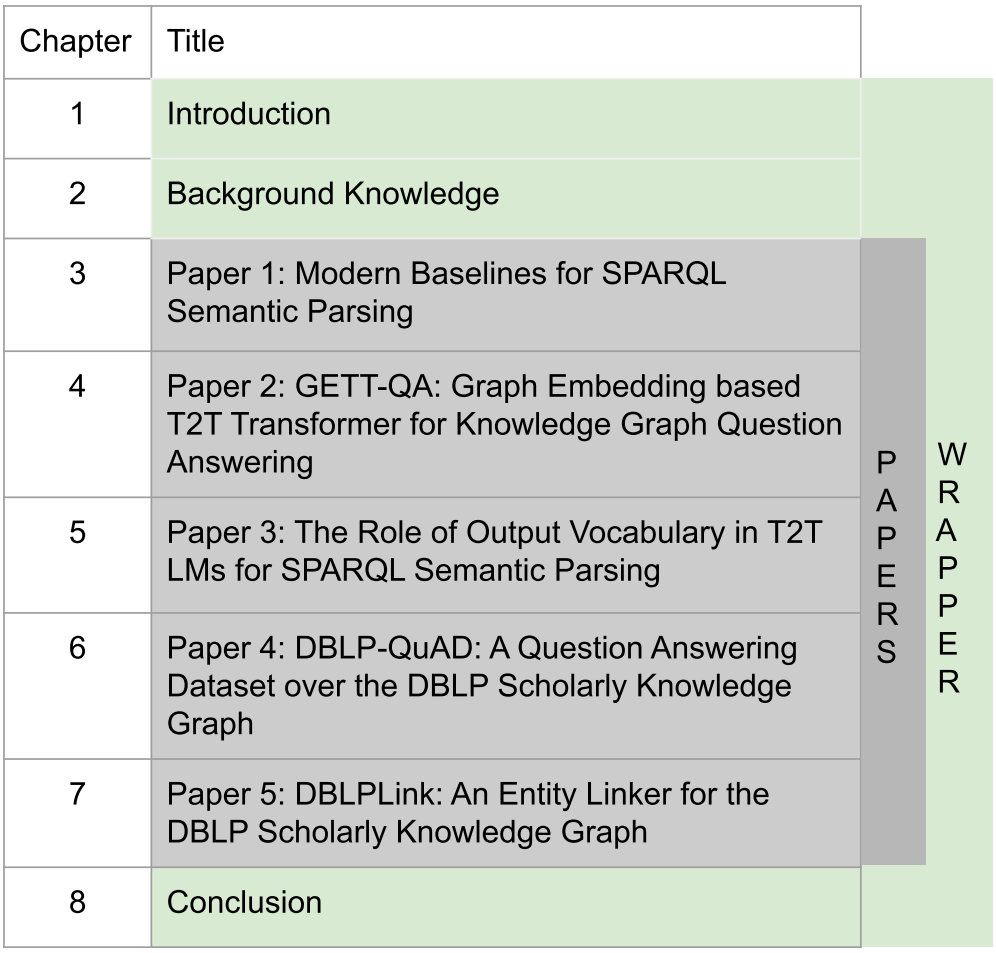


Figure 1.5: Thesis outline.

2
Background Knowledge

2.1 Introduction

The research presented in this thesis is performed on the shoulders of several fundamental pieces of technology, without which, conducting our experiments would not be possible. Without the seminal contributions of past researchers described in this chapter, barely any modern NLP system would exist on its own.

One of the most basic concepts in language is that of words, and how words represent meaning. Hence, we start by introducing word meaning representations commonly used in NLP. This aspect is relevant in an over-arching manner in our thesis, but specifically relevant to our work in Chapter 5, **The Role of Output Vocabulary in T2T LMs for SPARQL Semantic Parsing** (Banerjee, Nair, Usbeck, et al., 2023b). In this thesis, we experiment with modern sequence-to-sequence models, namely generative Language Models, on the task of semantic parsing. We start with the earliest sequence-to-sequence models, namely Recurrent Neural Networks (RNNs), and explain the workings of a popular variant, namely LSTMs (Hochreiter and Schmidhuber, 1997). We briefly discuss a special architecture called Pointer Generator Network (PGN) (See et al., 2017), which uses LSTM as an underlying technology. We make use of PGNs as a baseline model in our work **Modern Baselines for SPARQL Semantic Parsing** (Banerjee, Nair, Kaur, et al., 2022) which is a part of Chapter 3. Later, we dive deeper into pre-trained and Transformer-based (Vaswani et al., 2017) sequence-to-sequence models, which are precisely the focus of our thesis. Initially, we explain the Transformer architecture itself, and how it solves some serious problems faced by RNN and LSTM models. Subsequently, we describe T5 (Raffel et al., 2020) and BART (M Lewis et al., 2020) models, which are our generative LMs of choice throughout the thesis. We do not describe Knowledge Graphs as a concept any further than what has already been described in the previous chapter, in Section 1.1, however, we describe some KG embeddings in Section 2.7, addressing relevant work in Chapters 3 **Modern Baselines for SPARQL Semantic Parsing**, 4 **GETT-QA: Graph Embedding based T2T Transformer for Knowledge Graph Question Answering** and 7 **DBLPLink: An Entity Linker for the DBLP Scholarly Knowledge Graph**.

2.2 Adaptation Disclosure

The following sections have largely been adapted from the book "Speech and Language Processing" by Dan Jurafsky and James H. Martin (Jurafsky and Martin, 2009). The text has been modified for simplicity where suitable, while equations remain as they are in the original text. The figures have been re-drawn by me digitally, and simplified where possible. The section about Pointer Generator Networks (Section 2.5.5) has been written with reference to the original paper by See et al. (2017), however the diagram of a PGN is simplified, and the equations have been modified to appear uniform to earlier sections on sequence-to-sequence models in this chapter. The sub-sections for T5 and BART (sub-sections 2.6.3, 2.6.3) have been derived from their original respective papers, Raffel et al. (2020) and M Lewis et al. (2020).

2.3 Word Embeddings

A new research idea cooks best in my head when I am cooking.

In the sentence above, the two instances of *cook* mean different things. In the first case, cook means "develops" or "forms", while the second cook means the preparation of food in a kitchen. The same sentence could also be replaced with synonym phrases:

A new research idea forms in my head best when I am preparing a meal.

Although we replaced the word cook with approximate synonyms, most of the other words in the sentence and their order remain the same. In reality, words gain meaning from the context in which they are presented. It has been seen that words that occur in similar contexts have similar meanings. This is called **distributional hypothesis** and was first discovered by linguistic researchers such as Joos (Joos, 1950), Harris (Harris, 1954), Firth (Firth, 1957).

Words	**-3**	**-2**	**-1**	**0**	**1**	**2**	**3**
Happy					×		
Sad	×						
Love						×	
Hate		×					
Strong						×	
Weak	×						

Table 2.1: Osgood's Semantic Differential Scale denoting words a score in the range -3,+3.
Source: Osgood et al. (1957)

It is interesting to think about how a word should be represented. One approach is to use results from the linguistic study of word meanings, e.g., word meanings in a dictionary or a list of antonyms or synonyms. One may also categorize words based on their *sense*, whether they are positive or negative or represent hope or despair. In Table 2.1, Osgood's (Osgood et al., 1957) work on semantic differential scale represented words on a -3,+3 scale, which placed words with similar meanings on similar positions on the scale. A more significant breakthrough was achieved by Osgood when he

identified three aspects of words, namely evaluation, potency, and activity. Words can be placed on a scale on each of these attributes, which implicitly provides them a point in a coordinate system, as shown in Figure 2.1. This led to the development of the field of *vector semantics*.

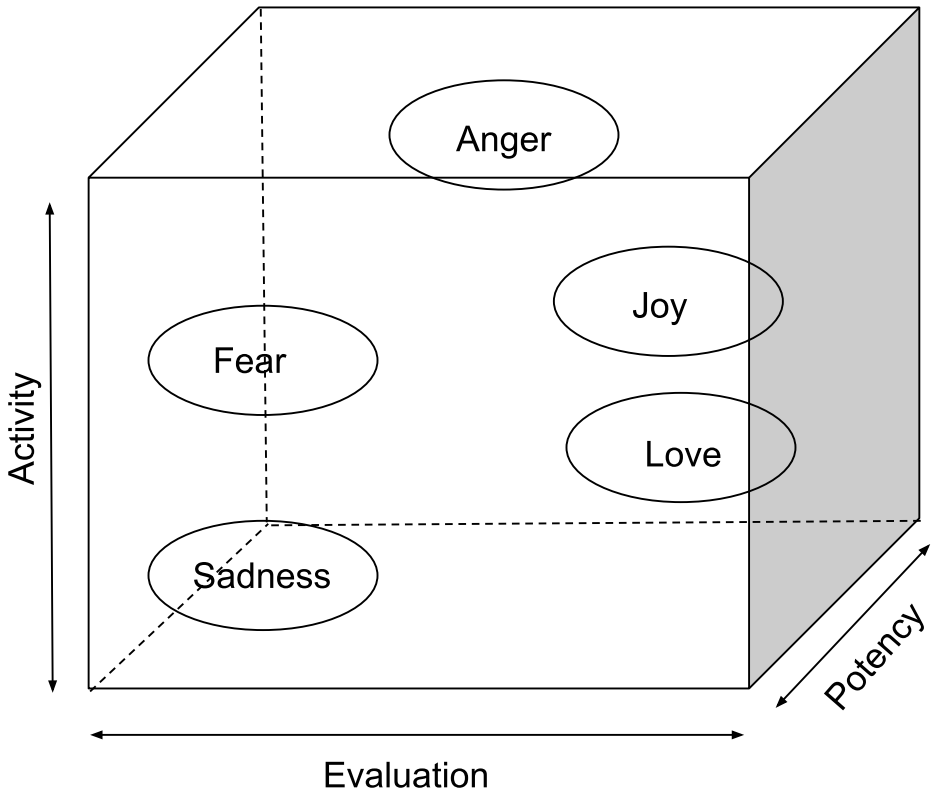


Figure 2.1: Osgood's representation of words based on three attributes.
Source: Osgood et al. (1957)

The problem with lexical semantics is that the initial classification of words into senses, and also the assignment of degree of membership to each sense, requires human involvement. As briefly discussed before, the idea of the **distributional hypothesis** comes to the rescue. Since similar words occurring in similar contexts probably mean the same thing, word representations can also be discovered automatically by mining text corpora. This brings us to *vector semantics*, which is a way to represent a word as a point in vector space. Similar words would reside closer to each other in this space, while words of the opposite meaning would reside the farthest from each other. Like every point in vector space, this word can be denoted by the coordinates of the vector space and can be written down as a series of numbers, or in other words, as a **vector**.

A simple approach to forming such vectors is to form a co-occurrence matrix of words. Let there be V words in the vocabulary, and let the matrix be a $V \times V$ matrix. Now, given a corpora, we count how many times a particular word was seen with another word. This leads to a matrix where each word is represented by a row of numbers. This row is the vector by which we denote this word. This is the basis for some popular vector approaches such as tf-idf (Salton and McGill, 1986). The problem with such approaches is that the matrix tends to be quite large in size. For example, if we have 5000 words in our vocabulary, the matrix would be of size 5000×5000. Moreover, the

length of each vector would be 5000 and would be sparse in nature since each word will only co-occur frequently with a small fraction of words in the large vocabulary.

It has been seen that shorter and dense vectors work better than larger and sparse vectors on a large variety of NLP tasks. The dense vectors are also called **embeddings**. The dimension of such embeddings does not depend directly on the size of the vocabulary and typically varies between 50 and 1000. One such popular dense embedding is Word2Vec Mikolov, Sutskever, et al. (2013).

In Word2Vec, a binary classifier is trained on the task of the prediction problem, whether a word W is likely to show up close to another word C or not. A corpus of text is considered as the training dataset, and as positive samples, a word W is taken out of a sentence, paired with the rest of the words in the sentence, and presented to the model for the probability that this word does indeed belong to the sentence. For negative examples, random words from the vocabulary are included instead of W. This method is called skip-gram, and in this method, no manual intervention is required for the preparation of training data and hence falls under the category of **self-supervision**. This class of word embeddings is called *static embeddings*, because the matrix produced by training, i.e., an embedding for each word, is used as it is in all downstream tasks. The embeddings for a given word have been decided to be fixed during the training process based on the corpora used. Since we do not make use of static word embeddings in our thesis, we limit our discussion to this point. For further details of Word2Vec, we refer the interested reader to Chapter 6 of the book "Speech and Language Processing" Jurafsky and Martin (2009).

As shown in a sample sentence at the start of this section, multiple instances of the same word may have different meanings in a given sentence, based on the context. Static embeddings fail to capture this nuance since each word is represented by a single embedding once training ends. Thanks to recent advances in the Transformer model, it is now possible to compute different representations for the same word given different contexts. In this thesis, we make widespread use of such embeddings. We shall come back to the topic of dynamic embeddings later in later subsections, once we have explained the Transformer model. Before we can explain the Transformer model, we must first explain more basic models, such as neural networks, feed-forward networks, and recurrent neural networks.

2.4 Neural Networks

The basic unit of a neural network is a neuron. As shown in Figure 2.2, given a set of inputs $x_1, ..., x_n$, a set of corresponding weights $w_1, ..., w_n$ and a bias b, the weighted sum z is:

$$z = b + \sum_i w_i x_i \tag{2.1}$$

A simpler way of writing the equation is as a dot product of the vectors w and x:

$$\mathbf{z} = \mathbf{w} \cdot \mathbf{x} + b \tag{2.2}$$

Usually, a non-linear function is applied on z so that a stacked layer of neurons can approximate a large family of functions. One popular non-linear function used is the sigmoid function:

$$y = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.3}$$

The sigmoid function has the nice properties that it produces an output in the range (0,1), and also diminishes outliers towards 0 or 1, and is also differentiable.

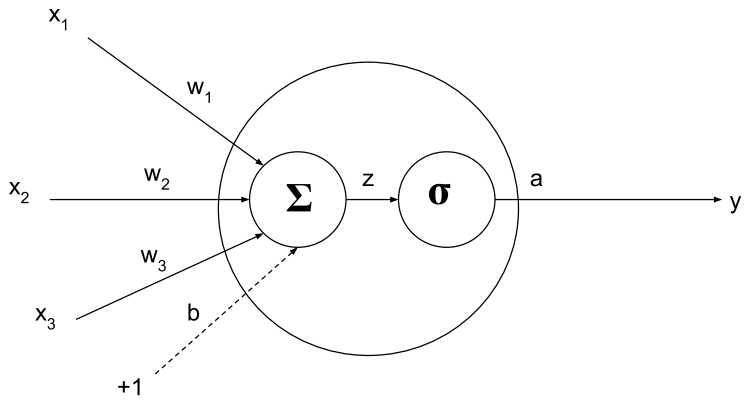


Figure 2.2: A Neuron. Inputs are weighted, summed and passed through the sigmoid function to introduce non-linearity.
Source: Jurafsky and Martin (2009)

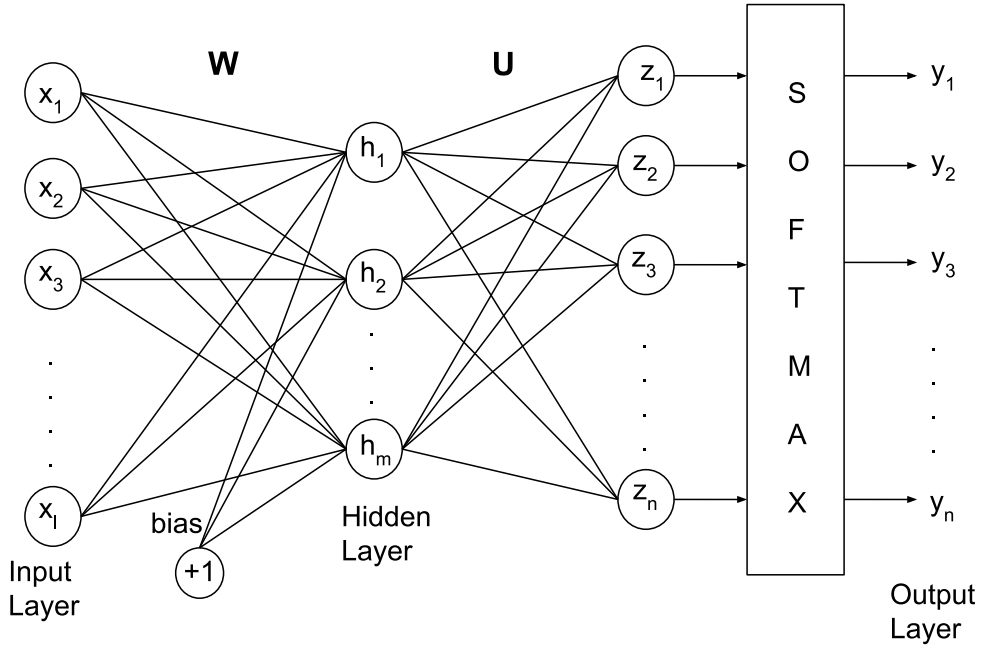


Figure 2.3: A Neural Network model. Multiple neurons and weights are arranged in layers, where the inputs undergo mathematical operations and progress from left to right during inference.
Source: Jurafsky and Martin (2009)

The simplest kind of neural network is called a **feed-forward neural network**, as seen in Figure 2.3. It is a multi-layer network where the output of one layer is carried over to the next. The outputs travel in only one direction, and not in loops or backwards.

A feed-forward neural network contains one of more **hidden layers**, which are built of individual neurons as shown in Figure 2.2. Normally, feedforward neural networks are fully connected, i.e., every pair of units in adjacent layers is connected.

The output of the hidden layer may be written as

$$h = \sigma(WX + b) \tag{2.4}$$

where W is the weight matrix lying between the input and hidden layers, while b is the bias matrix added to the hidden layer from the inputs. X is the input value in the form of a matrix. We apply the sigmoid function in the hidden layer.

The weights between the hidden layer and output layer are represented in Figure 2.3 as U. The output from the hidden layer is multiplied by these weights and produces an intermediate output Z:

$$\mathbf{Z} = U\mathbf{H} \tag{2.5}$$

z is a vector of real values, and can not normally used to make classification decisions. As a result, a special function called the **softmax** is applied to this vector to reduce it to a sequence of probabilities, where the sequence sums to 1.

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{d} e^{z_j}}, \quad 1 \leq i \leq d \tag{2.6}$$

where z is a vector of dimensionality d. This effectively produces the final output:

$$\mathbf{Y} = \text{softmax}(\mathbf{Z}) \tag{2.7}$$

2.5 Sequence to Sequence Models

Sequence-to-sequence models play a vital role in natural language processing because many tasks involve variable-length input and output sequences, such as language translation, summarization, and text generation. In our case, the focus on semantic parsing, a variant of text generation, requires a special discussion on this family of models. These models are often constructed with recurrent neural networks or Transformers.

2.5.1 RNN

Recurrent Neural Networks (RNNs) (Mikolov, Karafiát, et al., 2010) include a mechanism specifically designed for handling the sequential characteristics of language. This enables them to manage the temporal aspects of language without relying on pre-determined fixed-size windows. Through recurrent connections, the RNN introduces a novel method for representing the preceding context, enabling the model's decisions to be influenced by information spanning hundreds of words in the past.

In Figure 2.4, we see a basic RNN unit, with input vector x_t producing an output y_t, computed via the hidden layer h. This is similar to how a feed-forward neural network functions, however, in this case, the subscript **t** refers to the instances of input, output, and hidden layer output at a particular *point of time*. As the next input token x_{t+1} arrives, the previous hidden layer output h_t is combined with it to compute the next hidden

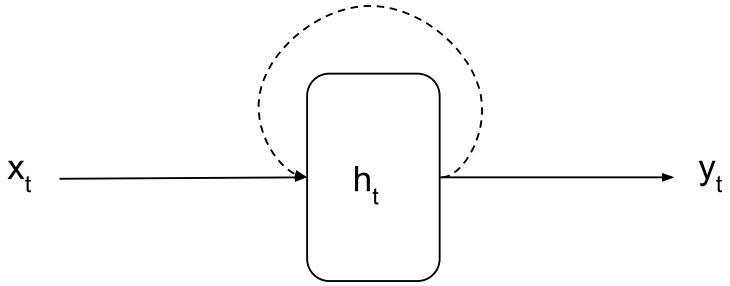


Figure 2.4: A Recurrent Neural Network model. The hidden state h_t is re-used for the computation of time step $t + 1$.
Source: Jurafsky and Martin (2009)

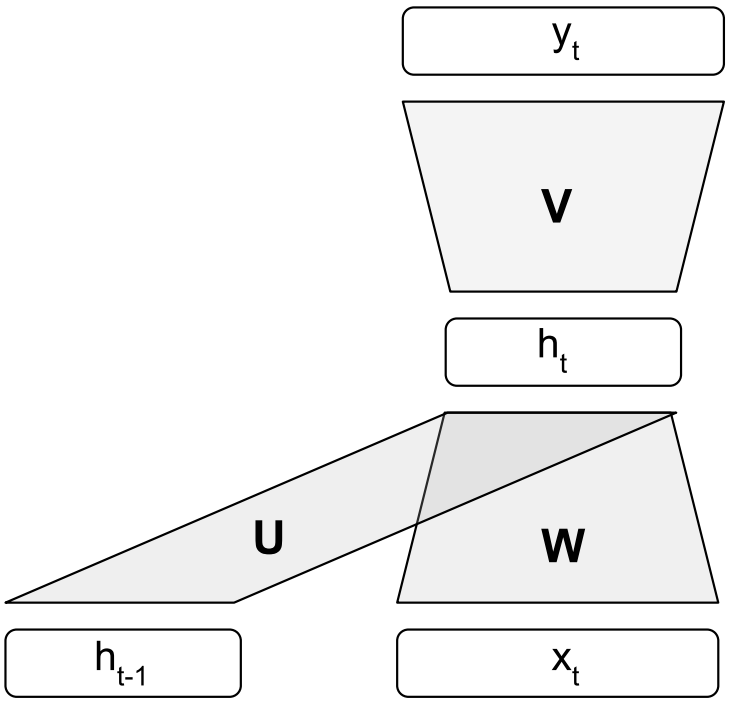


Figure 2.5: One Recurrent Neural Network time step.
Source: Jurafsky and Martin (2009)

layer representation. This introduces the concept of a "memory", where the previous hidden layer outputs carry forward the content of previous input tokens.

In Figure 2.5, we see the inference procedure of a single timestep for an RNN. The output vector y_t is produced from the input vector x_t by the following equations:

$$h_t = g(Uh_{t-1} + Wx_t) \tag{2.8}$$

$$y_t = f(Vh_t) \tag{2.9}$$

where U, W and V are weight matrices and h_t is the output of the hidden layer at timestep t. We may assume that g is the sigmoid function while f is the softmax function. The sequential nature of the computation requiring the combination of timesteps $t - 1$

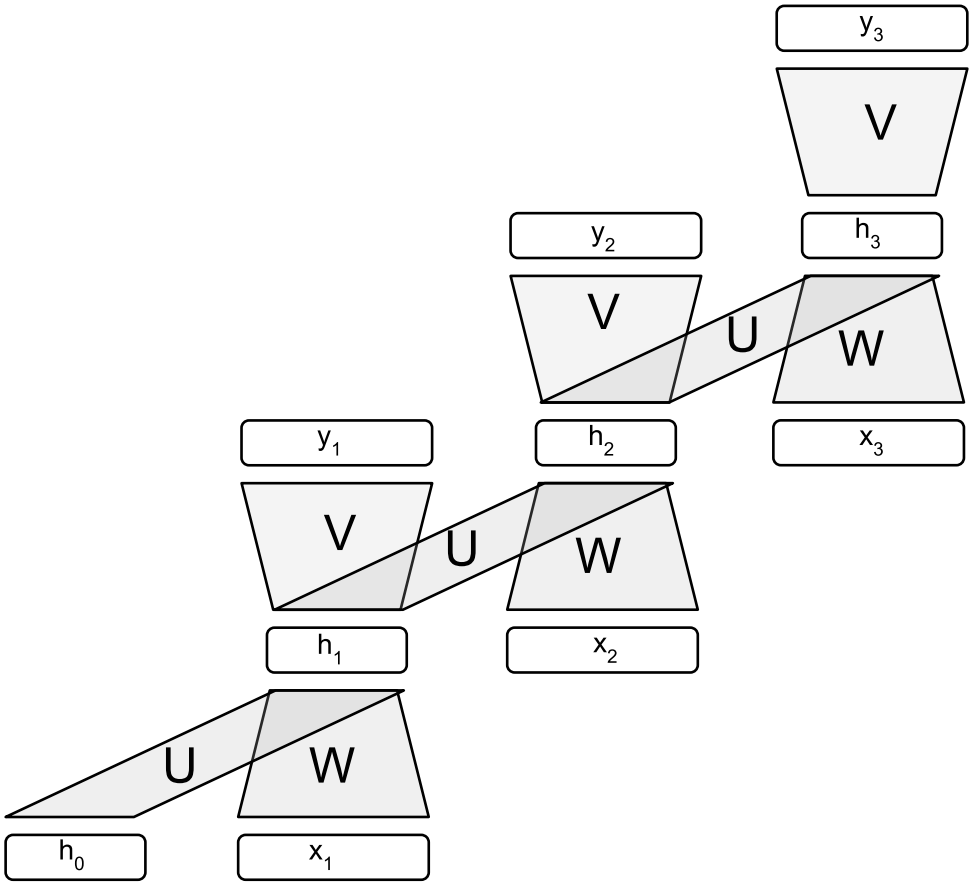


Figure 2.6: A Recurrent Neural Network "unrolled" to show computations across three timesteps. Source: Jurafsky and Martin (2009)

and t may better be visualized in Figure 2.6 where the *unrolled* network is shown performing the computations for three consecutive input tokens.

2.5.2 LSTM

RNNs are hard to train effectively because of the **vanishing gradients** problem. During training, fractional gradients in later steps in the sequence diminish quickly in the back-propagation process. Thus, they do not reach earlier input signals, making it hard for the RNN to capture long-range dependencies. Gating-based architectures, such as the LSTM (Hochreiter and Schmidhuber, 1997) and the GRU (Cho et al., 2014) are designed to solve this deficiency.

The LSTM unit shown in Figure 2.7 achieves this by adding a new **context layer** c_t, and using *gates* for dividing the "memory" management problems into two sub-parts: 1. removing information not likely to be used in the future 2. adding information likely to be used in the future. This is achieved by the addition of a **forget gate** and **add gate**.

The role of the forget gate is to remove information from the context no longer deemed necessary. We first compute a mask using the previous hidden layer state h_{t-1} and the current input x:

$$f_t = \sigma(U_f h_{t-1} + W_f x_t) \tag{2.10}$$

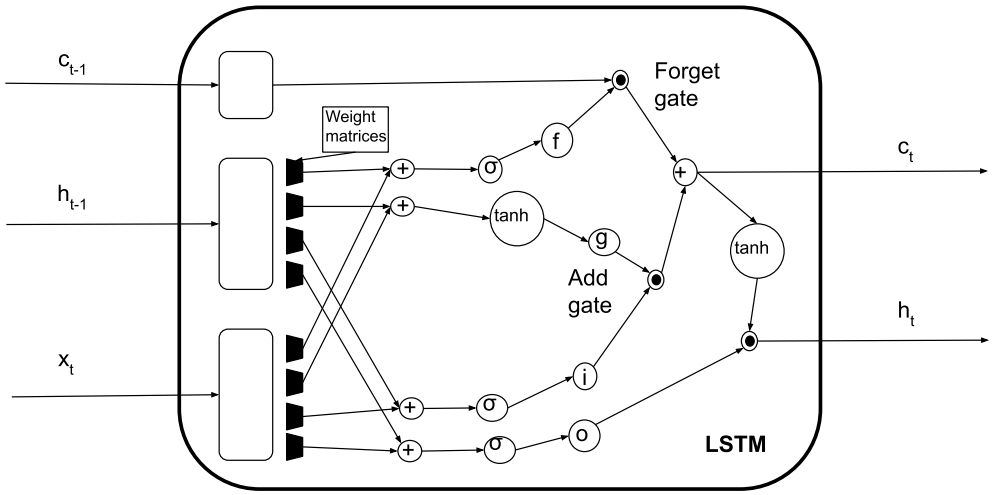


Figure 2.7: The Long Short-Term Memory (LSTM) architecture depicts the flow and computations of the input, the hidden state, and the control state.
Source: Hochreiter and Schmidhuber (1997)

Next, we do an element-wise multiplication of this mask to the context vector c_{t-1} to remove information no longer required:

$$k_t = c_{t-1} \odot f_t \tag{2.11}$$

Now that we have removed unnecessary information from the context, we focus on what information to add to the context to carry forward in timesteps. Using the previous hidden state and current inputs:

$$g_t = tanh(U_g h_{t-1} + W_g x_t) \tag{2.12}$$

We now generate a mask for the **add gate** to choose which parts of the information to add to the context:

$$i_t = \sigma(U_i h_{t-1} + W_i x_t) \tag{2.13}$$

$$j_t = g_t \odot i_t \tag{2.14}$$

We add the outputs of the add gate and forget gate to get the new context vector:

$$c_t = j_t + k_t \tag{2.15}$$

The new context vector is combined with the previous hidden state and the current input to generate the new hidden state:

$$o_t = \sigma(U_o h_{t-1} + W_o x_t) \tag{2.16}$$

$$h_t = o_t \odot tanh(c_t) \tag{2.17}$$

As shown above, the usual RNN components x_t, h_t are augmented by the context layer c_t, which through the add and forget gates, controls which information to propagate and which to diminish. This balance allows longer preservation of historical context during the LSTM operation.

2.5.3 Encoder-Decoder Model

As seen in Figure 2.8, the encoder-decoder model (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014) is used when the input sequence is of a different length than the output, and does not align in a word-to-word manner. Neural machine translation (e.g., translating from English to French) is a typical example where encoder-decoder models are used. Typically, an encoder-decoder model consists of two sets of RNNs, where one set is the encoder and the other is the decoder. The encoder takes as input a series of tokens and produces a fixed-length context vector c. The decoder then uses this context vector, along with the time-stepped output of its own, and produces the next token.

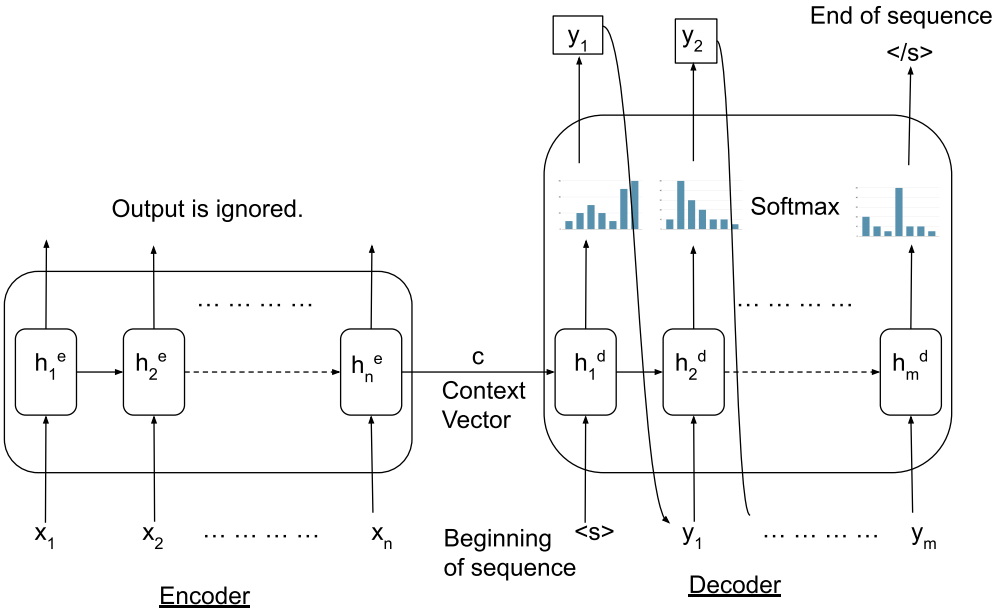


Figure 2.8: An encoder-decoder model. The context vector c relays the information from the encoder to the decoder.

Source: Jurafsky and Martin (2009)

In the most basic form of the encoder-decoder model, the context vector c is the last hidden state of the encoder and is passed as the initial hidden state of the RNN unit in the decoder.

$$c = h_n^e \tag{2.18}$$

$$h_0^d = c \tag{2.19}$$

In Figure 2.8, we have a more advanced version, where c is passed to each timestep of the decoder. This allows the decoder to repeatedly access the output of the encoder at each decoding step so that the influence of c does not diminish with each decoding step. Hence, for the decoder, the computation of the current hidden state is as follows:

$$h_t^d = g(y_{t-1}, h_{t-1}^d, c) \tag{2.20}$$

This hidden state is passed through an activation function f to a softmax function and the most likely word in the output vocabulary is produced as output.

$$z_t = f(h_t^d) \tag{2.21}$$

$$y_t = softmax(z_t) \tag{2.22}$$

Note from the above equations that the output at each decoding step y_t is passed as input to the next decoding step. This is combined with the previous decoder's hidden state and the encoder context c to compute the next hidden state and output. This phenomenon of passing the output of one step to the other step is called **auto-regressive** and is a typical nature of several models used to process language.

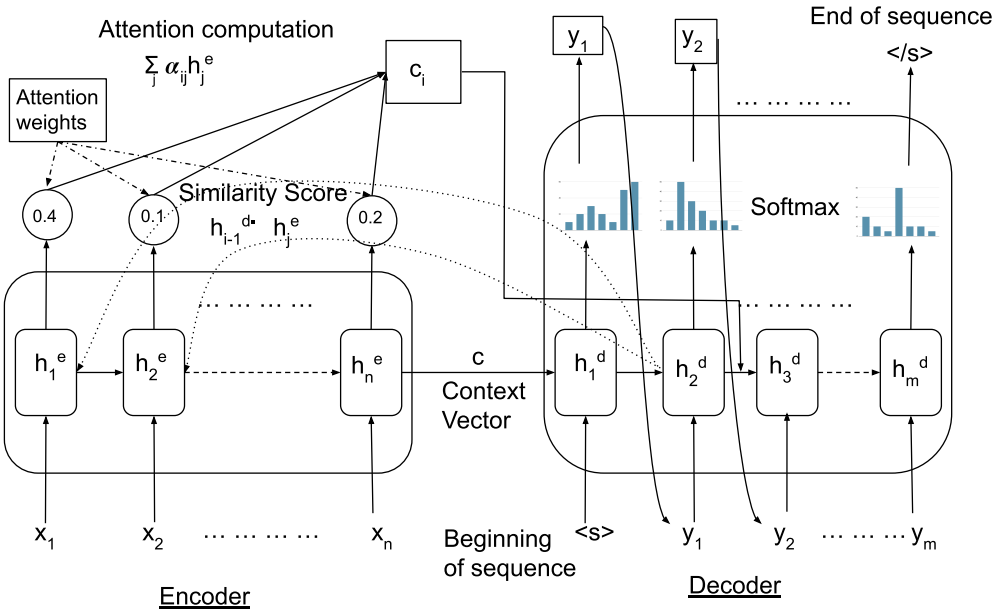


Figure 2.9: An encoder-decoder model with attention. The decoder is no longer reliant solely on c, but can also rely on the attention weights for transmission of information from encoder to decoder.

Source: Jurafsky and Martin (2009)

2.5.4 Attention

In encoder-decoder models, the context vector c produced by the encoder must contain the entire substance of the input to be eventually transmitted to the decoder. Due to the fixed size of the context vector, it becomes a **bottleneck**. Typically, for longer sequences, the earlier tokens no longer find themselves significantly represented in c. In Figure 2.9 (Bahdanau et al., n.d.; Graves, 2013), we depict an encoder-decoder with **attention mechanism**. Instead of relying on a single context vector c, the decoder has access to c_i, which is a new context vector for decoding timestep i. c_i is computed as a weighted sum of all the hidden states of the encoder. To compute the weighting, a score of similarity is computed at each decoding timestep i between the decoder hidden state h_{i-1}^d and the encoder hidden state at state j, h_j^e. We may use the dot-product here for attention computation:

$$score(h_{i-1}^d, h_j^e) = h_{i-1}^d \cdot h_j^e \tag{2.23}$$

This score represents the degree of similarity between the hidden vectors of the respective encoder and decoder hidden states. Naturally, the encoder state, which has higher similarity, should weigh higher in the computation of c_i. This is done by applying a softmax function over the scores and summing the weighted context vectors of each encoder hidden state:

$$\alpha_{ij} = softmax(h_{i-1}^d \cdot h_j^e) \tag{2.24}$$

$$c_i = \sum_j \alpha_{ij} h_j^e \tag{2.25}$$

This produces a fixed-length context vector that can now be used by each step of the decoder. It has been seen that encoder-decoder models with attention mechanisms outperform vanilla encoder-decoder models by large margins on most tasks.

2.5.5 Pointer Generator Network

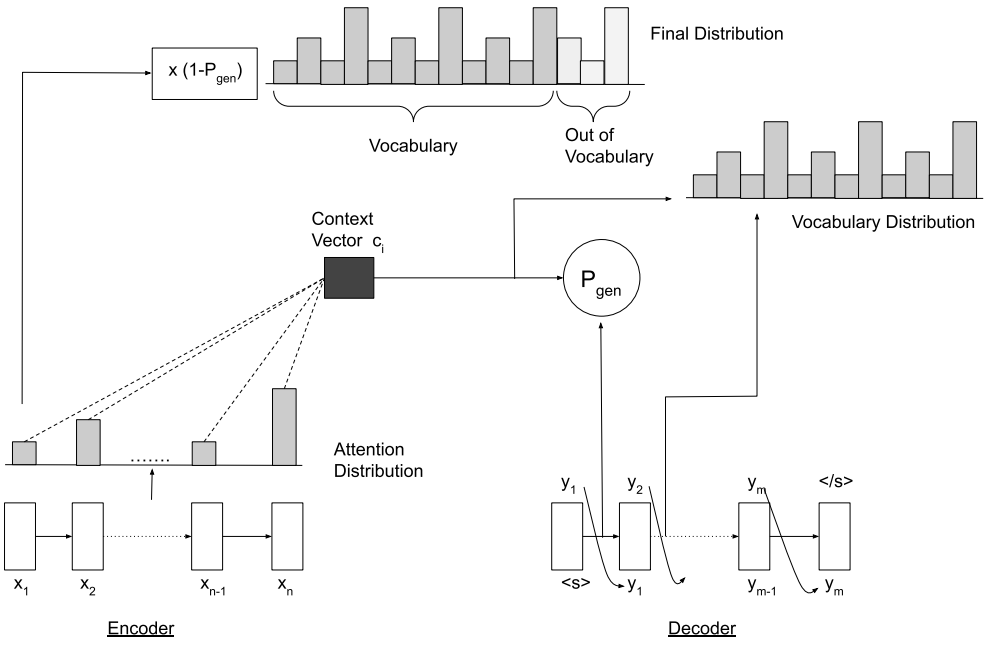


Figure 2.10: A Pointer Generator Network. This encoder-decoder architecture allows the production of out-of-vocabulary items due to its "copying" ability.
Source: See et al. (2017)

In our work **Modern Baselines for SPARQL Semantic Parsing** (Banerjee, Nair, Kaur, et al., 2022), we use a modified architecture of the encoder-decoder with attention model, called the Pointer Generator Network (PGN) (See et al., 2017). As seen in Figure 2.10, the encoder, decoder, and context vectors are similar to what has been seen before. However, in the case of PGN, the output not only produces a softmax distribution over a fixed vocabulary, it is also able to produce a distribution over the input tokens. Some of the

input tokens may be out-of-vocabulary, and this kind of model allows the production of tokens in output that were never seen before during training.

At the core of the PGN lies the generation probability p_{gen}, which controls, at each decoding step, whether a token is chosen from the vocabulary or from the input tokens, which may also include out-of-vocabulary tokens. The computation of p_{gen} is carried out as:

$$p_{\text{gen}} = \sigma(Uc_i + Vh_i^d + Wy_i + b_{ptr}) \tag{2.26}$$

Where U, V, and W are learnable weight vectors and b_{ptr} is a learnable scalar parameter. c_i is the context vector at timestep i, h_i^d is the hidden state of the decoder at timestep i and y_i is the output of decoder at timestep i.

Next, p_{gen} is used as a soft-switch to choose between generating a word from the vocabulary P_{vocab} or copying a word from the input sequence by sampling from the attention distribution α. For each document, let the extended vocabulary be denoted as the union of the vocabulary (e.g., all English words) and the words appearing in the document (including non-English words). The following distribution is computed over the extended vocabulary:

$$P(w) = p_{gen}P_{vocab}(w) + (1 - p_{gen})\alpha \tag{2.27}$$

In the context of KGQA semantic parsing, this model allows the copying of input tokens to the output query generated. This is required when a part of the input question must be copied verbatim in the produced SPARQL query. Examples of such queries may be found in Chapter 3.

2.5.6 Cross-Entropy Loss

When training models on language learning tasks, the loss function most often used is the cross-entropy loss. It is commonly employed as a loss function during training to assess how well a model's predicted probabilities align with the actual target labels.

Consider a binary classification task where the model predicts the probability of an instance x belonging to a specific class (denoted as class 1). The target label y can be either 1 (for class 1) or 0 (for class not 1). The cross-entropy loss for a single data point can be expressed as:

$$H(y, \hat{y}) = -[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})] \tag{2.28}$$

where $H(y, \hat{y})$ represents the cross-entropy loss between the true label y and the predicted probability $\hat{y}$, log denotes the logarithm (natural logarithm in this case).

The term $y \log(\hat{y})$ measures the penalty for incorrectly predicting class 1 when the true label is 1 (i.e., the high predicted probability for the wrong class). Similarly, $(1 - y) \log(1 - \hat{y})$ penalizes the model for predicting class 1 with low probability when the true label is, in fact, 1 (i.e., low predicted probability for the correct class).

The concept extends to multi-class classification problems as well. Let's assume there are C possible classes and the model outputs a vector of probabilities $\hat{y} = [\hat{y_1}, \hat{y_2}, ..., \hat{y_C}]$ where each element $\hat{y_i}$ represents the predicted probability for class i. The corresponding target label y would be a one-hot encoded vector with a 1 at the index corresponding to the true class.

The multi-class cross-entropy loss is then calculated as:

$$H(y, \hat{y}) = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{2.29}$$

Here, the sum iterates over all classes, and each term $y_i \log(\hat{y}_i)$ contributes to the overall loss based on the true label (y_i) and the predicted probability for that class ($\hat{y}_i$). During model training, the goal is to minimize the average cross-entropy loss over the entire training dataset. This is achieved using optimization algorithms that iteratively update the model's parameters to reduce the discrepancy between predicted probabilities and true labels. All the models discussed so far and also subsequently in this chapter are trained using the cross-entropy loss.

2.6 Transformers

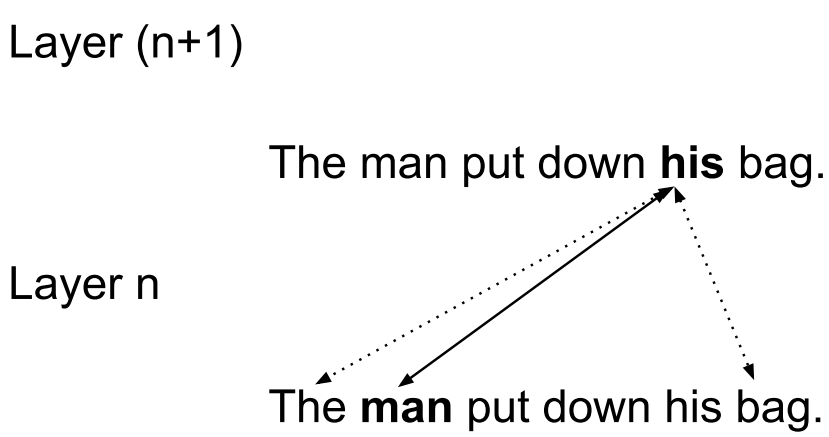


Figure 2.11: The concept of self-attention depicted through a sample sentence. Here, "his" attends most strongly to "man". Hence, the representation of "his" should carry higher weightage for the representation of "man".

Recurrent models face challenges in handling sequential data due to their inherent sequential processing, resulting in limitations in parallelization, difficulty capturing long-range dependencies, and susceptibility to gradient vanishing and exploding problems.

The Transformer (Vaswani et al., 2017) model addresses this issue by leveraging **self-attention** mechanism in an input text. Self-attention is a concept derived from the closely related concept of attention presented in earlier sections. As seen in Figure 2.11, self-attention is a mechanism for building a contextual representation of words based on neighboring words. In the given example, the word "his" refers closely to "man", and hence the contextual representation for "his" would have a higher weightage of the contextual representation of "man". This also means that for the given word "his", the contextual representation would change in a different sentence. This also introduces the concept of dynamic word embeddings, which, as opposed to static word embeddings (e.g., Word2Vec), produce different representations for the same word in different contexts.

In Figure 2.12, we introduce the architecture of the self-attention mechanism within a transformer block. We start by introducing the concept of **query**, **key** and **value**.

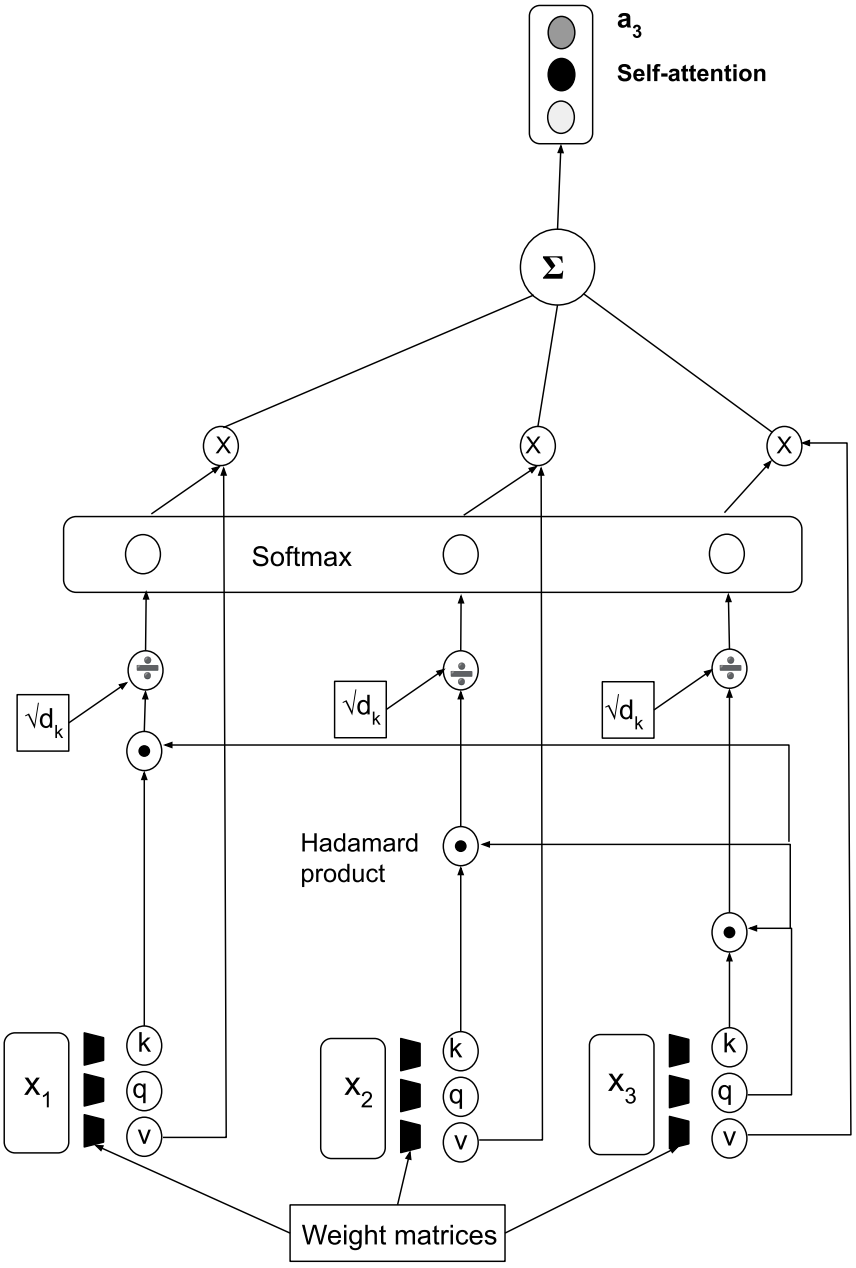


Figure 2.12: Self-attention mechanism in Transformer model. For the computation of $x3$'s representation, its query vector is compared against the key vectors of $x1$ and $x2$, and that is used as a weighting factor for combining the value vectors of all three.
Source: Jurafsky and Martin (2009)

Each input embedding to the transformer block plays three roles. In the query role, the embedding is the focus of attention and is compared to all the previous input tokens. In the key role, it is the preceding input being compared. In the value role, it is the computed output for the current focus of attention. To represent the three roles, the transformer introduces weight matrices W^Q, W^K, and W^V.

$$q_i = x_i W^Q \tag{2.30}$$

$$k_i = x_i W^K \tag{2.31}$$

$$v_i = x_i W^V \tag{2.32}$$

Thus, we have, for the input embedding x_i, the projected roles of query, key, and value. In reality, the dot products may result in extremely large values and lead to numerical instability during training. Hence, they are scaled down by dividing by a square root of the dimensionality of the query and key vectors, which in the original transformer paper is kept as 64.

Each word must be expressed as a weighted sum of the representation of words that came before it. The natural way to do this is to compare the word representations with each other with a dot product and use a softmax function over the result to create the weighting factor for the summation of the representations. In this case, the similarity between words at position i and j is computed as :

$$score(x_i, y_j) = q_i \cdot k_j \tag{2.33}$$

The softmax based weigthing calculation is performed as:

$$\alpha_{ij} = softmax(score(x_i, y_j)) \tag{2.34}$$

Here, we must note that $j \leq i$ is because of the causal nature of words in a sentence. As seen in Figure 2.13, a word is generally predicted based on the preceding words, not the subsequent words. Hence, for a focus word, we only compare it to words before it in the input sequence.

Given the proportional scores in α_{ij}, we compute final representation of x_i as a_i by summing the value vectors:

$$a_i = \sum_{j \leq i} \alpha_{ij} v_j \tag{2.35}$$

It is important to note that in the equations above, no computation depends on the results of an earlier computation. Although input token i is compared against input token j, the representations of i and j are taken as they are at the same timestep. This implies that the computation of representation for each input token can be done in parallel, which is, in fact, one of the most significant contributions of the transformer architecture. It is due to this property that transformers can be trained on a large corpus of text on special hardware carrying several parallel cores, giving rise to the paradigm of **pre-training** of language models.

Coming back to the topic of self-attention, the transformer model uses the concept of **multi-headed** self-attention. As seen in Figure 2.14, instead of computing just one representation for each input token, multiple attention heads compute their own representations of each token, which are then concatenated and passed through a new weight matrix to produce the final representation.

$$MultiHeadAttention(X) = (head1 \oplus head2 \oplus head3 \oplus head4)W^o \tag{2.36}$$

Here we assume that $\oplus$ represents concatenation, and there are four attention heads, where the new weight matrix is called W^o. The purpose of multi-head attention is to allow richer representations for a given input token. An input token may relate to other tokens in the input sentence in more than one way, and it is expected that among the multiple heads, these different forms of interactions are captured.

With self-attention defined, we can now describe the architecture of a single transformer block, as seen in Figure 2.15. For each input token, the multi-headed attention-based representation is computed. Each of these representations is passed through **layer norm**, which stands for layer normalization. Layer normalization is a technique that improves training performance by keeping the values of the hidden layer in a range that is appropriate for balancing gradients. The layer norm takes a single vector of dimension d as input and produces a vector d as output. First, the mean μ and standard deviation σ of the elements of the vector are computed. Given a hidden layer with dimensionality d_h:

$$\mu = \frac{1}{d_h}\sum_{i=1}^{d_h} x_i \tag{2.37}$$

$$\sigma = \sqrt{\frac{1}{d_h}\sum_{i=1}^{d_h}(x_i - \mu)^2} \tag{2.38}$$

A new vector is hence computed as:

$$\hat{x} = \frac{(x - \mu)}{\sigma} \tag{2.39}$$

In the standard implementation of layer norm, two learnable parameters γ and β are added:

$$LayerNorm = \gamma\hat{x} + \beta \tag{2.40}$$

In the transformer block in Figure 2.15, we see a **residual connection** passing the input vectors directly to the layer norm, by-passing the multi-head attention layer. It was shown by K He et al. (2016) that passing activation information forward and the gradient backward to skip a layer improves learning and gives later-level layers access to information from previous layers.

Apart from multi-headed attention, layer norm, and residual connection, the remaining component in a transformer block is the **feed-forward layer**. Each feed-forward layer is a 2 layer network, with individual point-wise networks for each input token. One token is passed through its own 2-layer network, and these weights are not used by or shared among other input tokens.

Putting it all together, the computations inside a transformer block are as follows:

$$O = LayerNorm(X + SelfAttention(X)) \tag{2.41}$$

$$H = LayerNorm(O + FFN(O)) \tag{2.42}$$

where H is the output of the transformer block. The output dimension and input dimension of the transformer block are kept the same so that multiple blocks can be stacked on top of each other. Large Language Models (LLMs) typically stack between 12 to 96 blocks to create greater learning capacity within the models.

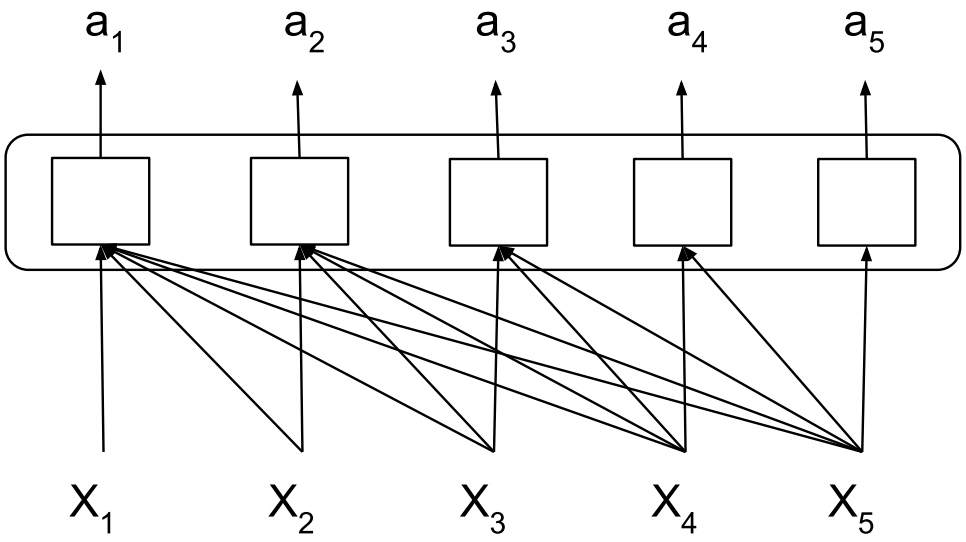


Figure 2.13: Causal self-attention. As an example, for the generation of token $X3$, the decoder attends to tokens that appeared before, i.e. $x1$ and $x2$.
Source: Jurafsky and Martin (2009)

2.6.1 Input Embeddings

Having described the entire Transformer architecture, we now briefly discuss the computation of the input embedding. Looking at the lower part of Figure 2.17, the input embedding consists of two parts: 1) the word embedding produced from the **embedding matrix E** and 2) **positional embedding**.

For each input sentence $S = w_1, w_2...w_n$ where w_i is a word in the sentence, a tokenizer such as BPE (Sennrich et al., 2016) or SentencePiece (Kudo and J Richardson, n.d.) breaks the sentence into sub-word tokens, transforming S to $S = t_1, t_2....t_m$. Note that there may be more tokens when compared to words because complex words may be broken into multiple parts by the sub-word tokenizer. Before training begins, the size of the token vocabulary is fixed at $|V|$. An embedding matrix of size $dX|V|$ is initialized, possibly with random values, where d is the size of the embedding (768 in BERT, to be introduced in the next section) and $|V|$ is the size of vocabulary (30522 for BERT). The sequential nature of language requires special attention to the position of each token in the input sentence. As such, apart from the static token embeddings initialized above, additional positional information must be embedded into the input embeddings. For this purpose, a combination of cosine functions with different frequencies was used in the original transformer paper, which represents the absolute position of tokens in a sequence. More complex positional embeddings for representing relative positions also exist. Eventually, the word embedding $E[token\_id]$ is summed with $PositionEmbedding(token\_position)$ and used as the input embedding. During pre-training, the embedding matrix E is updated and fine-tuned for optimal representation of input embeddings. Once pre-training is complete, fine-tuning for downstream tasks may further modify the embedding matrix for better performance.

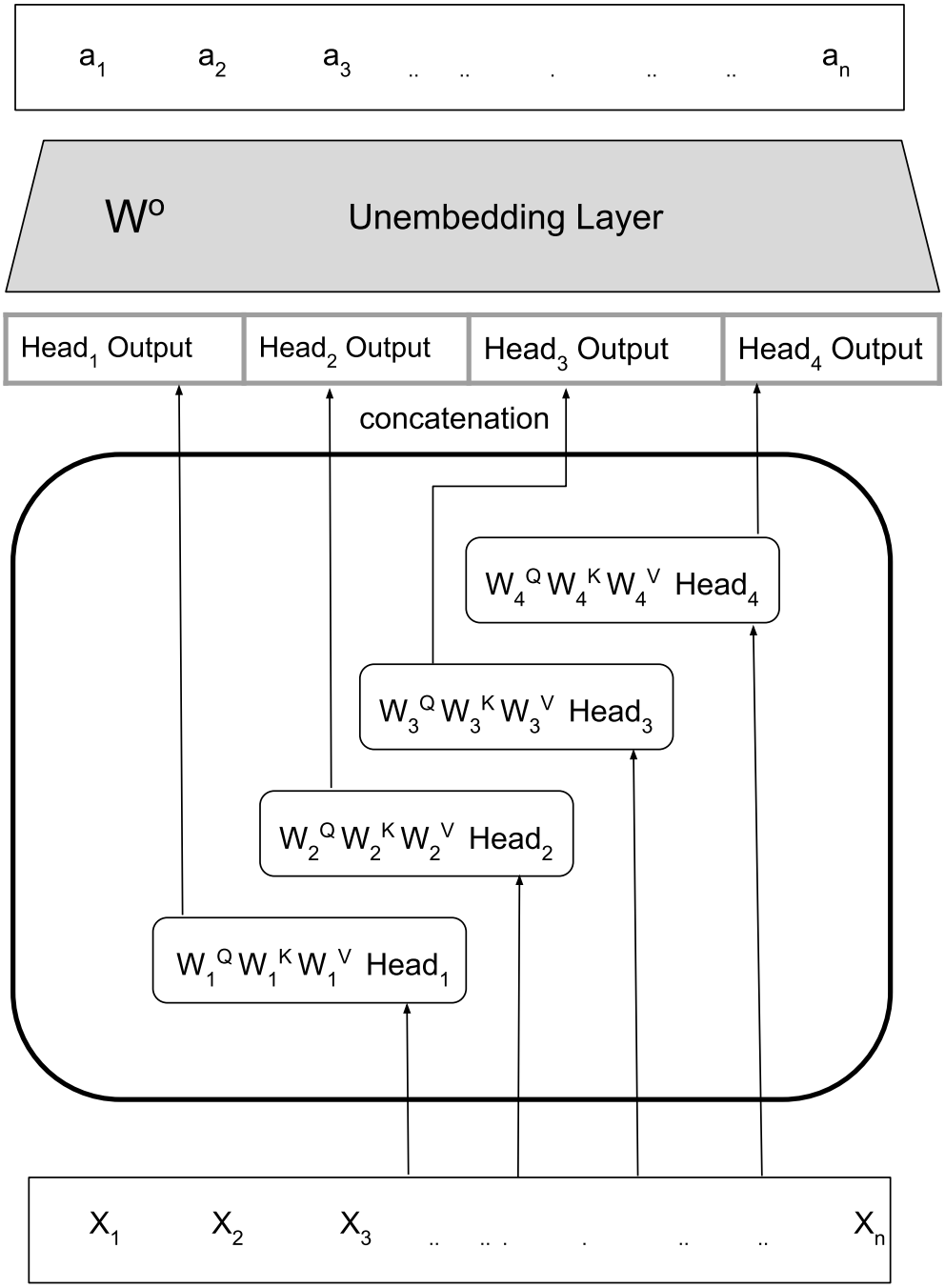


Figure 2.14: Multi-head self-attention mechanism in Transformer model. Each head produces its own representation for a given token, which is later concatenated to produce a richer representation than what a single head could have produced.

2.6.2 BERT

BERT (Devlin et al., 2019), which stands for Bidirectional Encoder Representations from Transformers, is a pre-trained language model based on the Transformer architecture. A key aspect of BERT is its pre-training phase. This phase involves training the model on

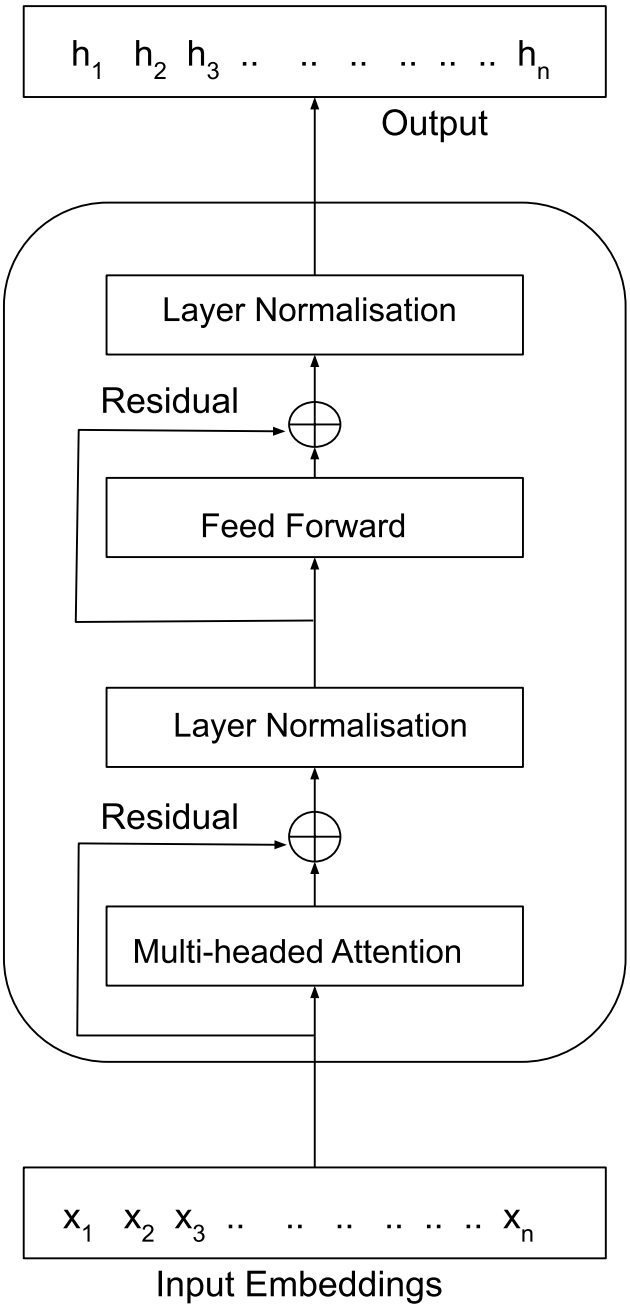


Figure 2.15: The Transformer block. Multi-headed attention is at the core of this model, which is further normalized and fed forward before producing representation for the given input.
Source: Vaswani et al. (2017) and Jurafsky and Martin (2009)

a massive unlabeled text corpus in an unsupervised manner. BERT leverages two tasks during pre-training:

In Masked Language Modeling (MLM), a random selection of words in the input sentence is replaced with a special "[MASK]" token. The model then attempts to predict the original masked words based on the surrounding context. This process allows BERT to learn deep contextual representations for each word in a sentence, considering both its preceding and following words.

$$\text{MLM} = \underset{w \in \mathcal{V}}{\text{argmax}} P(w|\text{Context}_{[MASK]}), \tag{2.43}$$

where w represents the masked word, $\mathcal{V}$ denotes the vocabulary of all possible words, and $Context_{[MASK]}$ represents the remaining sentence with the mask applied.

In addition to MLM, BERT also employs Next Sentence Prediction (NSP) for pre-training. Here, the model receives pairs of sentences as input and predicts whether the second sentence follows the first sentence in the original document. This objective helps BERT understand the relationships between sentences and how they flow logically.

By pre-training on massive amounts of text data using MLM and NSP, BERT gains a strong understanding of language context. This pre-trained model can then be fine-tuned for various Natural Language Processing (NLP) tasks, achieving state-of-the-art performance in tasks like question answering, sentiment analysis, and text summarization.

Even without further fine-tuning, the embedding output of a BERT model is used widely as the word embedding of choice in many modern applications. In this thesis, we make use of BERT embeddings as input vectors to our PGN model in Chapter 3. In Chapter 4, we use BERT embeddings of relation candidate labels for finding similarity and sorting. In Chapter 7, we use the BERT embedding of input questions and candidate entity labels for forming our input vectors.

2.6.3 Text-to-Text Models

BERT is not ideal for the task of text generation. BERT produces as output a set of embeddings that can be used for downstream tasks such as classification. Moreover, BERT uses bi-directional attention when computing the value of the masked token. On the contrary, when generating text, the current token relies on the previously generated tokens. As a result, text-to-text Transformer models either rely on a Transformer encoder-decoder (e.g., T5, BART) model or purely on a transformer decoder model (e.g., GPT-2). In this thesis, we focus on encoder-decoder models and restrict our discussion to T5 and BART, which are also encoder-decoder models for text generation.

In Figure 2.16, we see a basic encoder-decoder architecture using Transformer blocks. Both T5 and BART use the same architecture. The encoder block contains the same components as described earlier in Section 2.6. However, the decoder contains two new components: 1. **cross-attention layer** and 2. **causal self-attention**. Cross-attention behaves similarly to multi-headed self-attention, except that while the queries still come from the previous layer of the decoder, the keys and values come from the output of the encoder. Causal-self attention, as in Figure 2.13, only attends to tokens generated in the previous timesteps. This is especially required in the decoder because decoding happens in an auto-regressive fashion, token by token.

The final output of the encoder $H^{enc} = h_1, ..., h_n$ is multipled by the cross-attention layer's weights W^K and value weights W^V, while the output of the previous decoder layer $H^{dec[i-1]}$ is multiplied by the cross-attention layer's query weights W^Q:

$$Q = W^Q H^{dec[i-1]} \tag{2.44}$$

$$K = W^K H^{enc} \tag{2.45}$$

$$V = W^V H^{enc} \tag{2.46}$$

$$CrossAttention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.47}$$

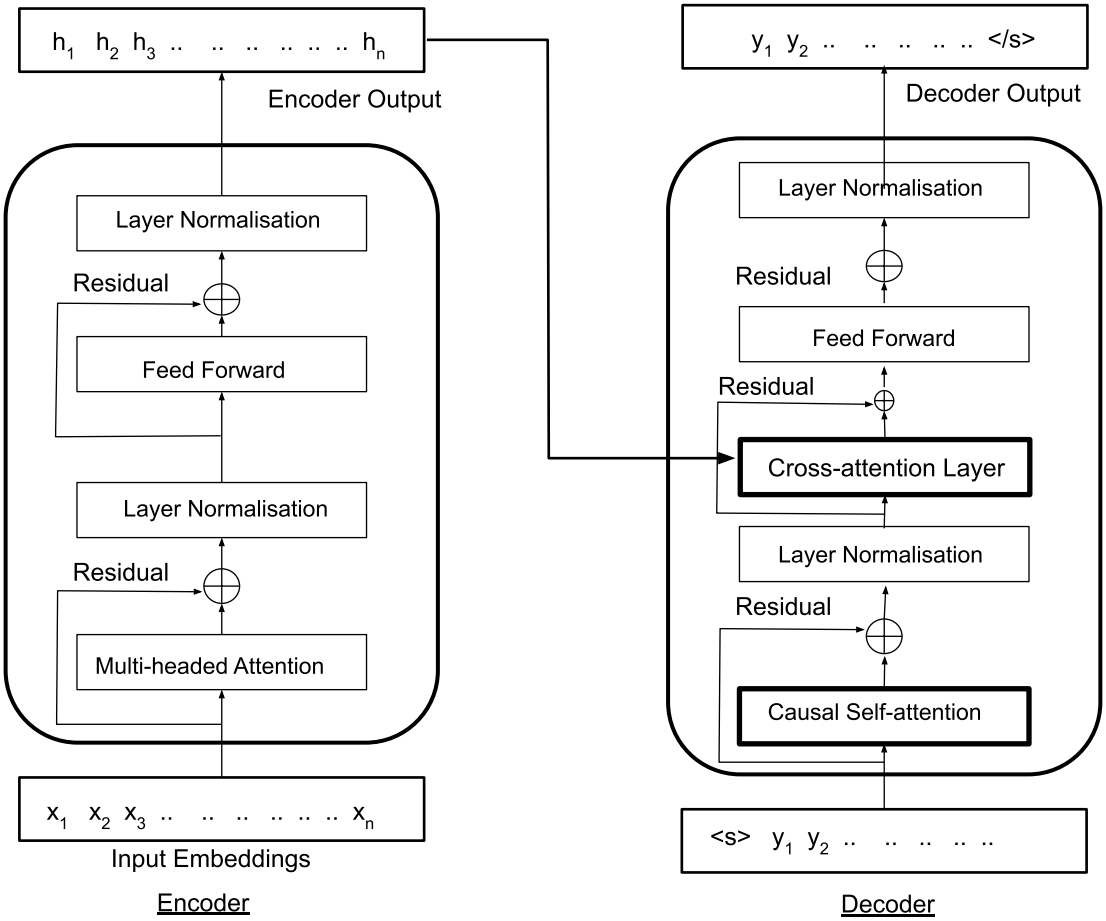


Figure 2.16: Transformer-based encoder-decoder model. In the decoder, the causal self-attention only attends to previously produced tokens, while the cross-attention layer attends to the encoder output.

Source: Jurafsky and Martin (2009)

The generation of text from these models requires further explanation. As seen in Figure 2.17, we depict a single transformer decoder block that takes X as input and produces H as output. Instead of the single Transformer block, one may also imagine a single-encoder-decoder model with the same inputs and outputs. Given the output embedding $H = h_1, h_2..., h_n$, the last embedding output h_n is passed through an **unembedding matrix**, which is of the inverse shape as that of the **embedding matrix E**, previously introduced in Subsection 2.6.1, used to generate input vectors. These matrices are **weight-tied**, i.e., during training, the same weight matrix is used at the input and the output but with reversed roles.

The unembedding layer produces output in the shape $1X|V|$ for each output item h_i, where $|V|$ is the size of the input and output vocabulary. These logit values are converted

to a probability distribution using the softmax function over the vocabulary. In the simplest sampling procedure, i.e., the **greedy approach**, the token with the highest probability value is selected for production, and the final text token y_i is produced. This is how the transformer model produces text.

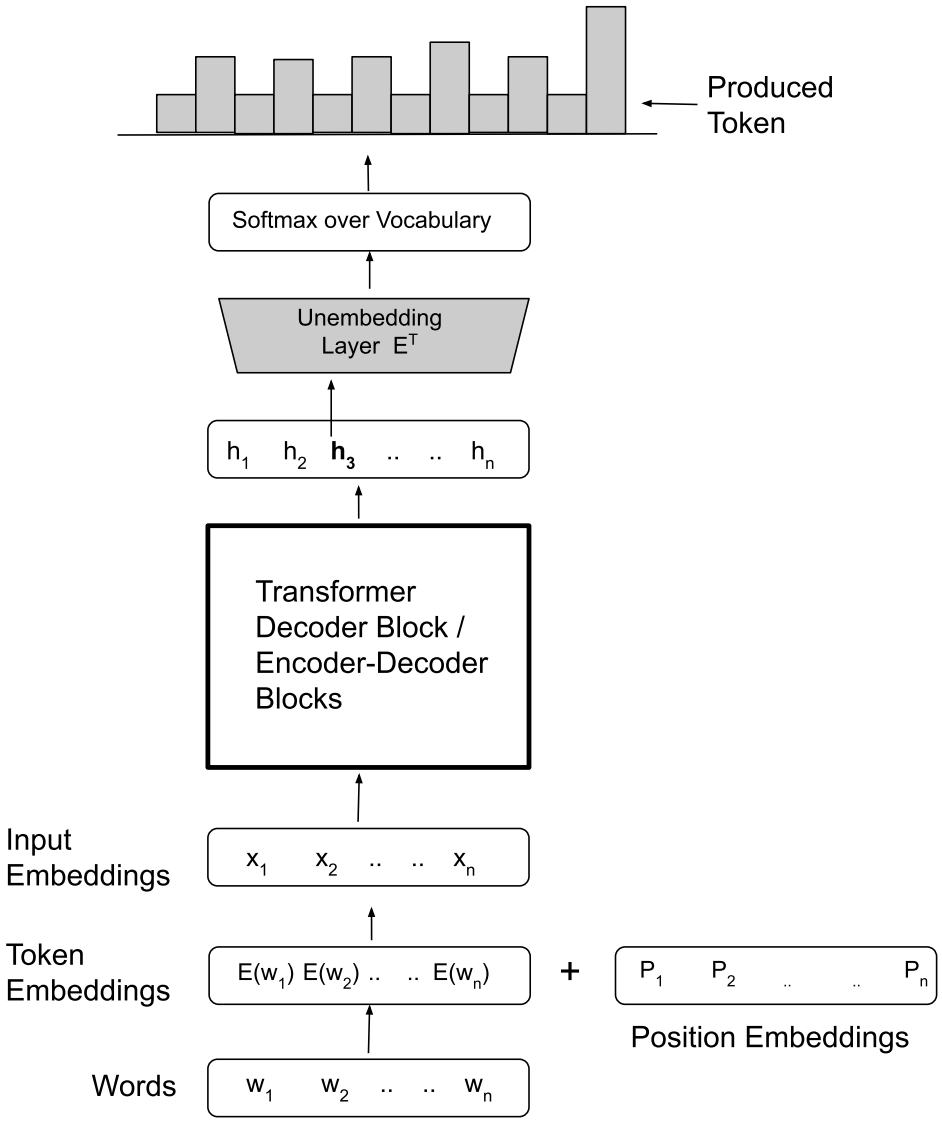


Figure 2.17: Text production from a Transformer model. In the beginning, token embeddings are combined with position embeddings. In the end, the embedding matrix E is transposed and used for generating a distribution over the tokens to be produced as output, which are combined to produce words.

Source: Jurafsky and Martin (2009)

T5

T5 Raffel et al. (2020), as seen in Figure 2.18, adopts a text-to-text approach for various tasks, such as translation, question answering, and classification. The methodology involves inputting text into the model and training it to generate corresponding target text. This approach facilitates the utilization of the same model, loss function, hyper-

parameters, and so on across a diverse range of tasks. T5 model has three popular variants: `T5-small` containing 60 million parameters, `T5-base` containing 220 million parameters, and `T5-large` containing 770 million parameters. In this thesis, due to computing constraints, we restrict our experiments to `T5-small` and `T5-base`. The C4 corpus, utilized for pre-training the models, has an uncompressed size of 750 GB. We make extensive use of the T5 model in this thesis, as it appears in Chapters 3, 4, 5, 6 and 7.

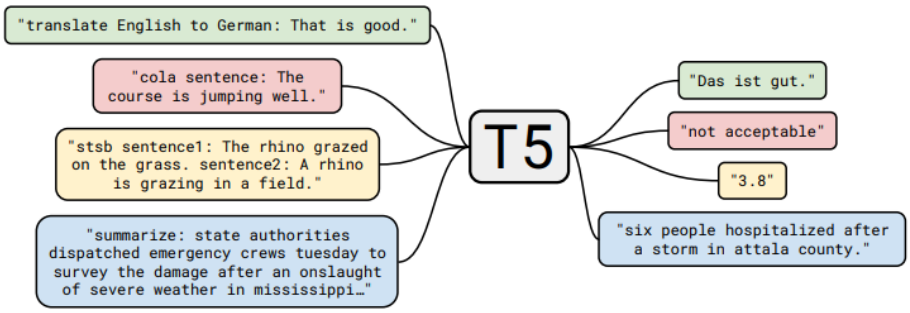


Figure 2.18: A diagram of the text-to-text framework for T5. Based on different pre-fixes, the model is able to perform a variety of tasks.
Source: Raffel et al. (2020)

BART

BART (M Lewis et al., 2020), which stands for Bidirectional and Auto-Regressive Transformer, is also a text-to-text, sequence-to-sequence model. In this thesis, we use the `BART-base` model, which encompasses 139 million trainable parameters. This model underwent pre-training on a dataset comprising 160 GB of combined data from four different corpora: Bookcorpus (Y Zhu et al., 2015), CC-News (Mackenzie et al., 2020), OpenWebText [1], and Stories (Trinh and Le, 2018).

T5 and BART differ in the corpus they were trained on. Moreover, as seen in Figures 2.19 and 2.20, the pre-training objectives of reproducing corrupted text remain similar in both models. However, the means of corruption vary, where T5 primary masks tokens to be re-constructed. At the same time, BART also implements approaches like shuffling the order of tokens, deleting tokens instead of applying a visible mask, and text infilling.

We make use of BART in Chapter 3, where we find certain limitations with its performance on the task of semantic parsing and rely more on T5 for the rest of the chapters in this thesis.

2.7 Knowledge Graph Embeddings

Knowledge Graphs (KGs) represent entities and relationships between them in a structured format. KG Embeddings aim to learn low-dimensional vector representations for these entities and relations, allowing us to reason and perform various tasks on the knowledge graph. We use KG embeddings in this thesis extensively in Chapters 3, 4 and 7. We describe below the three embeddings that were used in this thesis.

[1] http://skylion007.github.io/OpenWebTextCorpus

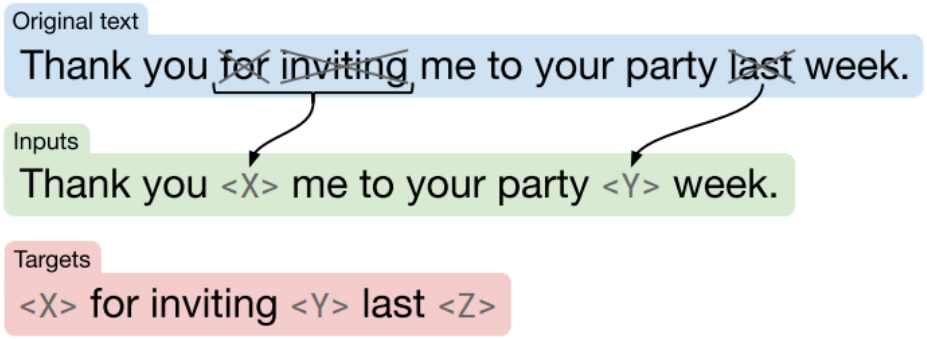


Figure 2.19: T5 pre-training framework for corrupting text. Words are deleted from the input text, and the task during pre-training is to predict the missing words.
Source: Raffel et al. (2020)

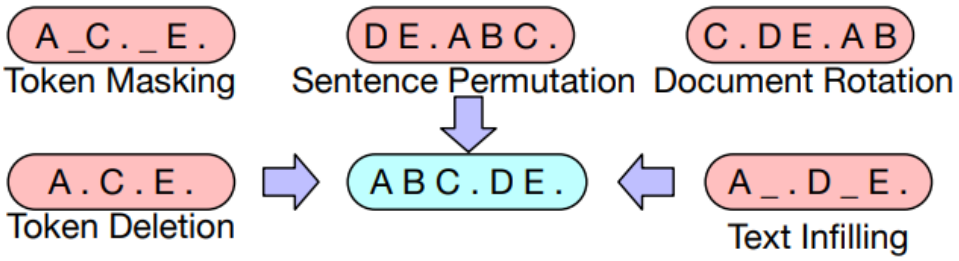


Figure 2.20: BART pre-training framework for corrupting the text. Compared to T5, BART uses more techniques for corrupting input text.
Source: M Lewis et al. (2020)

Let us assume entities are denoted as e and relations as r. The embedding function ϕ maps entities and relations to vector spaces:

- $\phi(e) \in \mathbb{R}^d$: Embedding vector for entity e in d-dimensional space.
- $\phi(r) \in \mathbb{R}^d$: Embedding vector for relation r in d-dimensional space.

The core idea behind KG embeddings is to capture relational patterns by encoding a certain mathematical property. For example, TransE (Bordes, Usunier, Garcia-Duran, et al., 2013) embeddings exploit the translation property denoted by the following relation:

$$\phi(h) + \phi(r) \approx \phi(t) \tag{2.48}$$

where:

- h - Head entity (subject)
- r - Relation
- t - Tail entity (object)

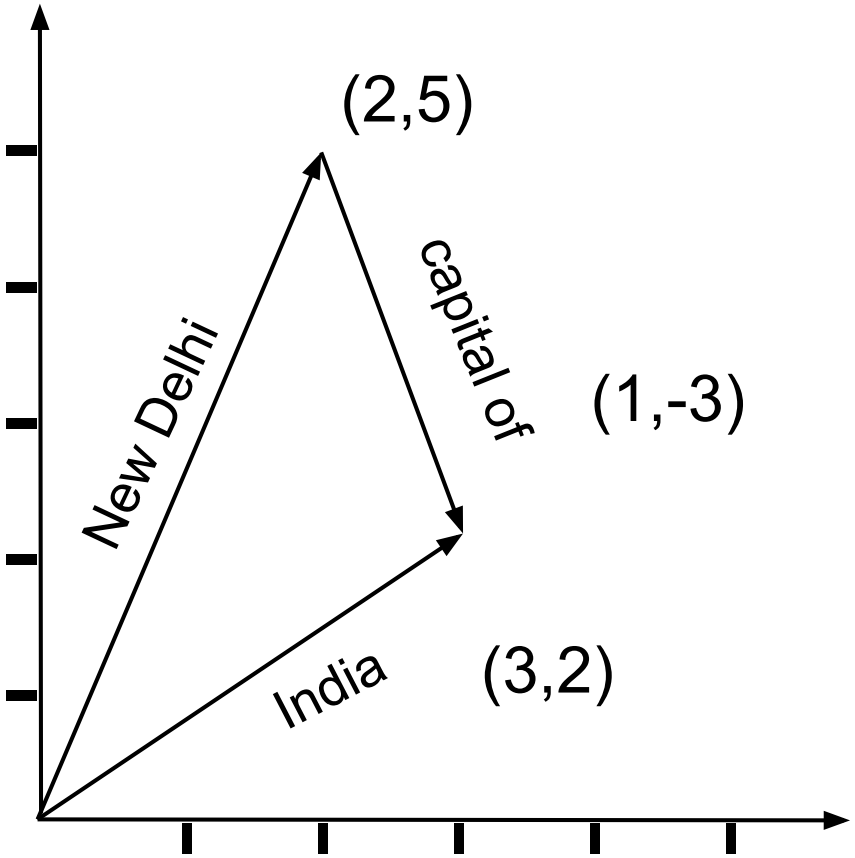


Figure 2.21: TransE KG embedding model. Entities and predicates are placed in vector space so that the relationship between them is reflected in the form of geometric mathematical operations.

As seen in Figure 2.21, this suggests that the vector representing the tail entity can be obtained by translating the head entity vector along the relation vector.

Consider a KG with entities Place (India, New Delhi) and relation "capital of". We can learn embedding vectors for these entities and the relation:

- ϕ(New Delhi) = [2, 5]
- ϕ(capital of) = [1, -3]
- ϕ(India) = [3,2]

According to the property, the sum of New Delhi's embedding and "capital of"'s embedding should be close to India's embedding:

$$\phi(\text{New Delhi}) + \phi(\text{capital of}) = [3, 2] \tag{2.49}$$

This closeness suggests that the model has learned a meaningful relationship between the entities and the relations. The scoring function for TransE may be written as:

$$f(h, r, t) = \|\phi(h) + \phi(r) - \phi(t)\|_2^2 \tag{2.50}$$

Here, $\|.\|_2$ denotes the L2 norm or Euclidean distance. A lower score indicates a better fit between the triplet (h, r, t), signifying that the translated head entity is closer to the actual tail entity in the embedding space.

Several KG embedding models implement the core principle with variations. For example, **DistMult** (Yang et al., 2015) moves away from the distance-based metric of TransE and instead computes the dot-product-based similarity of the head, relation, and tail vectors. It models thTransE can not express symmetric (e.g., roommate, neighbor) relations and 1-to-N relations (e.g., students of).r:

$$f(h, r, t) = \phi(h) \circ \phi(r) \circ \phi(t) \tag{2.51}$$

where $\circ$ denotes element-wise product. Intuitively, the score function can be viewed as a cosine similarity between $\phi(h) \cdot \phi(r)$ and $\phi(t)$. The higher the score, the better the alignment.

Both TransE and DistMult suffer from certain limitations. TransE can not express symmetric (e.g. roommate, neighbour) relations and 1-to-N relations (e.g. students of). DistMult is unable to express anti-symmetric relations (e.g., part-of, member-of) and inverse relations (e.g., father-of son-of).

To add greater expressivity in embedding spaces, **ComplEx** (Zhang et al., 2019) leverages complex-valued embeddings for entities and relations. It introduces separate real and imaginary components to model asymmetric relations, where the direction of the relation matters. The scoring function measures the dot product between the projected head and the conjugated tail:

$$f(h, r, t) = Re[conj(\phi(h)) \cdot \phi(r) \cdot \phi(t)] \tag{2.52}$$

Where Re denotes the real part of a complex number. and $conj(z)$ denotes the complex conjugate of z. ComplEx is able to express symmetric, anti-symmetric, inverse and 1-to-N relations. However, it fails to represent compositional relations (e.g. $a \rightarrow b$ and $b \rightarrow c$ implies $a \rightarrow c$), which TransE is able to handle.

Which KG embedding to use depends on the nature of relations in the KG being used. If the KG does not have many symmetry relations, TransE works well. On the other hand, if the KG has several anti-symmetric, inverse, or 1-to-N relations, ComplEx works best. However, ComplEx is more expensive to compute owing to its complex number space and additional computations. DistMult presents a much cheaper alternative if the KG contains mostly symmetric and 1-to-N relations.

For further information on the topic of KG embeddings, we refer the interested reader to Chapter 4 of the "Graph Representation and Learning Book" (Hamilton, n.d.).

3

Modern Baselines for SPARQL Semantic Parsing

Bibliographic Information

Debayan Banerjee, Pranav Ajit Nair, Jivat Neet Kaur, Ricardo Usbeck, and Chris Biemann. 2022. Modern Baselines for SPARQL Semantic Parsing. In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY, USA. Pages 2260–2265.
https://doi.org/10.1145/3477495.3531841

3.1 Abstract

In this work, we focus on the task of generating SPARQL queries from natural language questions, which can then be executed on Knowledge Graphs (KGs). We assume that gold entity and relations have been provided, and the remaining task is to arrange them in the right order along with SPARQL vocabulary, and input tokens to produce the correct SPARQL query. Pre-trained Language Models (PLMs) have not been explored in depth on this task so far, so we experiment with BART, T5 and PGNs (Pointer Generator Networks) with BERT embeddings, looking for new baselines in the PLM era for this task, on DBpedia and Wikidata KGs. We show that T5 requires special input tokenisation, but produces state of the art performance on LC-QuAD 1.0 and LC-QuAD 2.0 datasets, and outperforms task-specific models from previous works. Moreover, the methods enable semantic parsing for questions where a part of the input needs to be copied to the output query, thus enabling a new paradigm in KG semantic parsing. Code and data used for this work can be found at https://github.com/debayan/sigir2022-sparqlbaselines.

3.2 Introduction

Knowledge Graph Question Answering (KGQA) is the task of finding answers to questions posed in natural language, using triples present in a KG. Typically the following steps are followed in KGQA: 1) Objects of interest in the natural language question (NLQ) are detected and linked to the KG in a step called entity linking (EL). 2) The relation between the objects is discovered and linked to the KG in a step called relation linking (RL). 3) A formal query, usually SPARQL, is formed with the linked entities and relations. The query is executed on the KG to fetch the answer. This step is called Query Building (QB), and is the focus of this paper. We assume that gold entities and relations are made available, and our task is to generate the correct SPARQL query. We experiment with a Pointer Generator Network (See et al., 2017) with special vectorisation, and with more recent pre-trained language models such as T5 (Raffel et al., 2020) and BART (M Lewis et al., 2020). We choose models that are able to copy tokens from the input text to the output SPARQL query, apart from being able to handle normal category of questions. Questions which require copy operations are found in the dataset LC-QuAD 2.0 (Dubey, Banerjee, Abdelkawi, et al., 2019), for example:

Is it true that an Olympic-size swimming pool's operating temperature is equal to 22.4 ?

which has the corresponding SPARQL over Wikidata KG (Vrandečić and Krötzsch, 2014):

```
ASK WHERE
{
  wd:Q2084454 wdt:P5066 ?obj
  filter(?obj = 22.4)
}
```

We found no previous work for SPARQL QB that can handle such questions. We also show that on datasets where copying is not required (LC-QuAD 1.0 (Trivedi et al., 2017)), the approaches still exhibit a strong performance.

Our contribution is as follows:

- We find that with the correct input tokenisation, T5 outperforms all previous works and achieves state of the art performance on LC-QuAD 1.0 over DBpedia (Lehmann, Isele, et al., 2015), and LC-QuAD 2.0 over Wikidata.

3.3 Related Work

Diefenbach, Lopez, et al. (2017) studied several QA systems and concluded that the QB step is generally intertwined with rest of the modules in a pipeline, and hence not evaluated separately. Later, with the advent of Frankenstein (Singh, Lytra, et al., 2018) and Qanary (Singh, Both, et al., 2016), which are frameworks that allow modular construction of an end-to-end KGQA pipeline, it was seen that the QB module is the least researched aspect of the pipeline. Soon after, Singh, Radhakrishna, et al. (2018) presented a comprehensive survey of individual components of KGQA pipeline and evaluated Sina (Shekarpour et al., 2015) and NLIWOD[1] as individual QB components.

[1] https://github.com/semantic-systems/NLIWOD/

Several recent works attempt to solve complex query building by constructing intermediate query representations, such as staged query graphs (Yih, MW Chang, et al., 2015; Hu et al., 2018; Luo et al., 2018; Y Chen et al., 2020; Lan and J Jiang, 2020) based on λ-calculus or skeleton structures (Y Sun et al., 2020). These systems have no built-in method of performing copy operations from input text to output query. The system based on skeleton structure requires manual annotation of queries to corresponding skeletal structures, apart from the final generated query. Some other solutions rely on templates (Abujabal et al., 2017; J Ding et al., 2019; Soru, Marx, Moussallem, et al., 2017), which generally have the natural limitation of the query only being limited to the templates chosen beforehand (Athreya et al., 2021; Vollmers et al., 2021). In the work of J Ding et al. (2019), the templates are discovered from the input training data, however, the implementation of the method of discovery of such templates, structures, and sub-structures in the queries is dataset specific. It is unclear if the same method of discovery can scale to all datasets.

Most systems that work over query graphs and use λ-calculus start with the assumption that entities are already linked, and the relations must still be found. However, with the availability of joint entity and relation linkers such as EARL (Dubey, Banerjee, Chaudhuri, et al., 2018) and Falcon (Sakor et al., 2019) , we can develop semantic parsers which focus only on query building and receive both entities and relations pre-linked.

For non-KG semantic parsing, PLMs have been evaluated recently with a focus on compositional generalisation (Shaw et al., 2021). For KG semantic parsing, the Compositional Freebase Questions (CFQ) (Keysers et al., 2020) dataset that is based on the Freebase KG (Bollacker et al., 2008), has reported results with non-pre-trained Transformer-based models, while subsequent works (Herzig et al., 2021; Furrer et al., 2020) over the same dataset have experimented with PLMs. This line of work may be considered closest to ours, however, their focus is on compositional generalisation, while our focus is on the ability to faithfully copy input tokens to the produced query. CFQ has no questions that require such copy operations. Moreover, since Freebase is no longer an active project, our focus is on DBPedia and Wikidata. By exploring two new datasets and corresponding KGs, we expand the scope of research.

Some recent work on Complex Temporal Question Answering (Jia et al., 2021) handled questions that require extraction of time stamps from the input question. They use task-specific temporal extraction tools such as SUTime (AX Chang and Manning, 2012) and HeidelTime (Strötgen and Gertz, 2010) for this purpose, however in the end, they do not form a SPARQL query, and instead attempt to find the correct answer directly through an entity re-ranking approach. Our efforts are instead on finding such flexible single models which can perform such extraction tasks along with the building of SPARQl queries.

The models we experiment with pose no structural or template-based constraints neither at an intermediate stage nor during decoding, and are able to produce any possible query structure since we generate the final query token by token. They require the input question, the linked entities and relations, and a fixed vocabulary of all possible SPARQL tokens. This allows them to operate on all possible datasets and KGs.

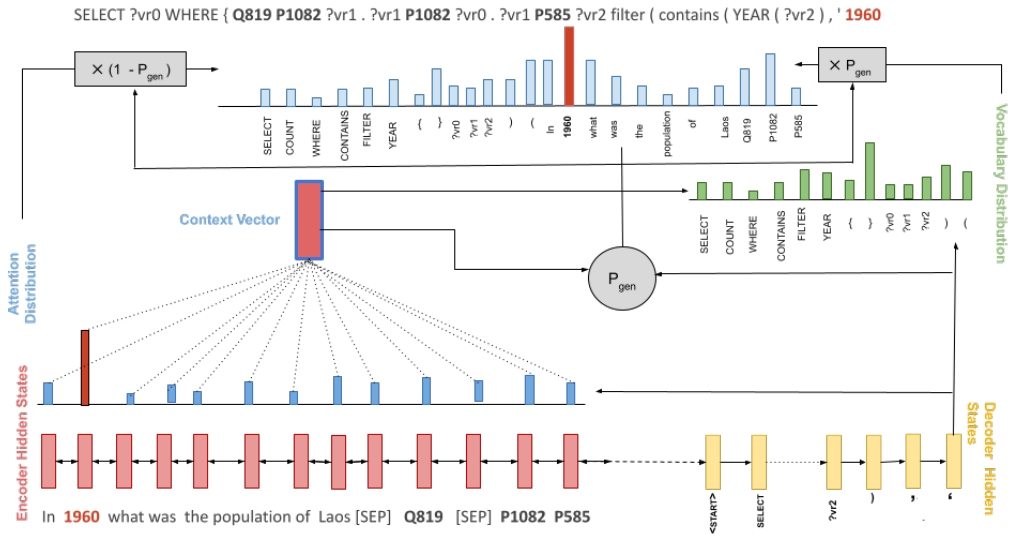


Figure 3.1: PGN-based QB model. At the current time step, the model is decoding the symbol after the single quote character ('). It considers the scores over the vocabulary and the attention weights over the input text to obtain a final probability distribution, from which it makes the prediction of choosing *1960* as the next token.

3.4 Models

3.4.1 T5

T5, or Text-to-Text Transfer Transformer, is a Transformer (Vaswani et al., 2017) based encoder-decoder architecture that uses a text-to-text approach. Every task, including translation, question answering, and classification, is cast as feeding the model text as input and training it to generate some target text. This allows for the use of the same model, loss function, hyper-parameters, etc. across a diverse set of tasks. We experiment with two variants of the T5 model: `T5-small` with 60 million parameters and `T5-base` with 220 million parameters. The uncompressed C4 corpus used to pre-train the models is 750 GB in size.

Input Tokenisation

Let the natural language input question be denoted by $Q = [w1, w2...wn]$ where w denotes the words in the question. The linked entities be denoted by $E = [E1, E2...En]$ and the relations by $R = [R1, R2....Rn]$. For each entity and relation we fetch the corresponding label from the respective KG such that E_{lab_1} denotes the label for E_1, and similarly for relations. Let the SPARQL vocabulary be donated by V. The input string to the model is formed as follows:

$$w_1 w_2 ... w_n [SEP] E_1 E_{lab_1} .... E_n E_{lab_n} [SEP] R_1 R_{lab_1} ... R_n R_{lab_n}$$

where each word is separated from the other word by a space. We denote the entities by their respective KG IRIs. For DBpedia the entities take the form, e.g. "`http://dbpedia.org/resource/Dolley_ Madison`" while the relations look like "`http://dbpedia.org/onto logy/spouse`". For Wikidata a typical entity looks like

wd:Q76 (Barack Obama) while a typical relation looks like wdt:P31 (instance of). The prefixes wd:, wdt: and several others are expanded according to the list available online[2]. We shuffle the order of entities and relations randomly before passing it to the model. When the input string is passed to T5 as it is, the accuracy is close to zero, as the AutoTokenizer splits the URLs as it sees fit, and it later fails to concatenate the fragments properly at the output. To work around this problem for DBpedia, we made use of the 100 sentinel tokens that T5 provides. The sentinel tokens were originally used during pre-training objective to denote masked tokens, but in our case, we make use of them to represent the prefixes, e.g, http://dbpedia.org/ontology/ as a specific sentinel token such as <extra_id_2>. Additionally, we represent each of the items in the SPARQL vocabulary V as a sentinel token with a specific ID for each keyword. We do the same for prefixes in Wikidata for LC-QuAD 2.0.

Output

The output is composed of tokens from Q, V, E and R. In the case of LC-QuAD 1.0, since there are no questions that require copying of tokens from Q, the output is composed of tokens from V, E and R. We would like to point out that since our input contains the entities and relations, the model always has to perform a form of "copying" even to decide where in the output SPARQL query to place the entities and relations. In the case of questions from LC-QuAD 2.0, in some cases this copying mechanism must also decide which input token from Q to copy and to which position in the output. The output for E and R are not produced verbatim, and instead they are split as a special token prefix, and then the entity or relation ID. A post-processing step is required to resolve these sentinel token IDs into the original prefixes and combine them with the adjacent IDs. For example, wdt:P31 is produced in the output as <extra_id_3> P31.

3.4.2 BART

The Bidirectional and Auto-Regressive Transformer or BART is a Transformer that combines the Bidirectional Encoder with an Autoregressive decoder into one Seq2Seq model. We experiment with the BART-base model consisting of 139 million trainable parameters. This model was pre-trained on 160 GB of data resulting from the combination of four corpuses, namely, Bookcorpus (Y Zhu et al., 2015), CC-News (Mackenzie et al., 2020), OpenWebText (Gokaslan and Cohen, 2019) and Stories (Trinh and Le, 2018).

Input Tokenisation

We form the input string for BART in the same manner as discussed in Section 3.4.1 for T5. However there is no concept of sentinel tokens in BART. Hence to handle special tokens, we instead add them to the BartTokenizer using the add_tokens function and appropriately resize the token embedding space using the resize_token_embeddings function.[3]

[2] https://www.wikidata.org/wiki/EntitySchema:E49
[3] https://huggingface.co/docs/transformers/model_doc/bart

3.4.3 Pointer Generator Network

A PGN is a seq2seq network with an encoder and a decoder block. We experiment exclusively with LSTM-based (Hochreiter and Schmidhuber, 1997) PGNs. While PGNs can generate output tokens from a given vocabulary via softmax selection, they are also able to copy tokens from the input text to the output by making use of attentions weights of the decoder. A significant difference between T5, BART and PGN is that for PGNs no pre-training on corpus is performed. The model consists of 53 million trainable parameters which is comparable to `T5-small`'s 60 million parameters size.

PGNs have typically been employed for abstract summarisation tasks but recently Rongali et al. (2020) used PGNs to perform SQL semantic parsing. Figure 3.1 depicts this architecture parsing an example query.

	F1	F1
	DBpedia 16.04	Wikidata
	LC-QuAD 1.0	LC-QuAD 2.0
Sina (Shekarpour et al., 2015)	0.24	-
NLIWOD	0.48	-
SQG (Zafar et al., 2018)	0.75	-
CompQA (Luo et al., 2018)	0.77	-
SubQG (J Ding et al., 2019)	0.85	-
AQG-net (Y Chen et al., 2020)	-	0.45
Multi-hop QGG (Lan and J Jiang, 2020)	-	0.53
CLC (Zou et al., 2021)	-	0.59
PGN-BERT	0.67	0.77
PGN-BERT-BERT	0.88	0.86
BART	0.84	0.64
T5-Small	0.90	**0.92**
T5-Base	**0.91**	0.91

Table 3.1: Results for query generation with gold entities and relations. Best results are in bold.

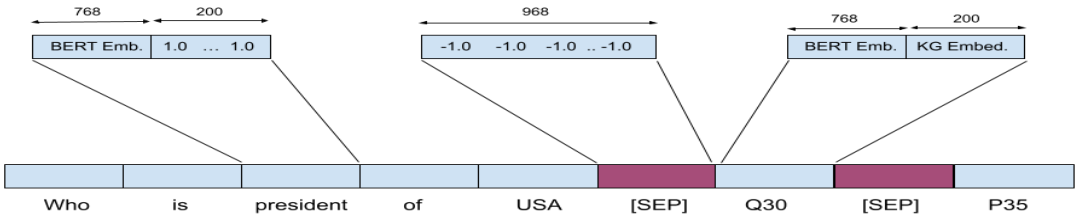


Figure 3.2: Input vector for PGN-BERT.

Input Tokenisation

We construct our input vector (Figure 3.2) by concatenating the question tokens with gold entities and relations. We add a separator token `[SEP]` between question tokens and entity candidates, as well as between entity candidates and relation candidates to help the model understand the segments of the input tokens. Each token is represented by a 968-dimensional vector, where the first 768 dimensions are the BERT contextual embeddings of the question token, or the entity label or relation label, as the case may

be. The next 200 dimensions are reserved for the KG embeddings. For question tokens, KG embeddings do not apply, so we fill the 200 dimensions with a sequence of `1.0`. The `[SEP]` token is represented by a vector full of `-1.0`. The linked entities and relations carry the respective KG embeddings in the last 200 dimensions.

KG Embeddings

For both datasets, we make us of TransE embeddings, due to its popularity and ease of availability (Bordes, Usunier, Garcia-Duran, et al., 2013). For DBpedia, We train TransE embeddings ourselves using PyTorch-BigGraph (Lerer et al., 2019). For the training, we proceed to 30 epochs and use the cosine dot operator as the comparator function. For Wikidata we made use of readily available TransE embeddings made available by Pytorch-BigGraph[4] on their webpage.

Re-Ranker

We found that the beams produced by the PGN often contained the right query but not in the top position. To improve the ranking of queries further, we fine-tuned a pre-trained BERT-based classifier (`distil-bert-base-uncased`) on the output of the PGN. We take the top-10 beam outputs of the PGN on the val sets and query the KG. We save the output of all the queries which produce a valid response from the KG. To train the BERT-based classifier, we form the input string by concatenating the question tokens, the SPARQL query, and the KG response. For supervision, we provide binary labels 0 and 1 for right and wrong query. Once trained, we use the logit values of the penultimate layer of the model for re-ranking the queries produced by PGN on the test set.

3.5 Datasets

- **LC-QuAD 1.0** contains 5,000 questions that are answerable on DBpedia 2016-04. The dataset originally contains a 4:1 train-test split. However, several competing systems in Table 1 instead use a smaller split of 3,253 questions. We adopt the same split and follow J Ding et al. (2019) in performing 5-fold cross-validation with 70:10:20 split for train, dev, and test, respectively. There are no questions that require copying of input tokens to output query in this dataset.

- **LC-QuAD 2.0** is based on Wikidata and consists of a mixture of simple and complex questions that were verbalized by human workers on Amazon Mechanical Turk. It is a large and varied dataset comprising 24,180 train questions and 6,046 test questions. In addition, to the best of our knowledge, this is the sole dataset that incorporates aspects of hyper-relational (Galkin et al., 2020) structure of recent Wikidata versions. Approximately 16% of questions in the dataset require copying of input tokens to the output query.

[4] https://github.com/facebookresearch/PyTorch-BigGraph

3.6 Evaluation

In Table 3.1, the results of competing systems for LC-QuAD 1.0 are shown as reported by J Ding et al. (2019). The results for competing systems for LC-QuAD 2.0 are as reported by Zou et al. (2021), also found on the KGQA leaderboard (Perevalov et al., 2022)[5]. For evaluating T5, BART and PGNs on both datasets, we take the top-10 beams outputs of the models and query the KG in ranked serial order. The first query to return a non-empty response is considered the output of the model. We match the KG responses of the generated query to the gold query and mark it a match if they are identical, and no match if they are not.

In the case of LC-QuAD 2.0, we found a large number of queries no longer answerable on the current version of Wikidata. The original dump used to create the dataset is no longer available online[6]. As a result, we setup an endpoint with the dump dated 13 October 2021[7] of Wikidata and filtered the test set of 6046 questions down to 4211 questions for which the gold query produced a valid response. We loaded the triples to a Virtuoso Open Source Triple Store and used it as our endpoint. Zou et al. (2021) resort to a similar pre-filtration step for the dataset in their work.

In Table 3.1, `PGN-BERT` refers to the setting where we feed the PGN with BERT embeddings. We apply no re-ranker to the output beams, and consider the first query in the ranked beams to produce a valid response from the KG as the output of the system. On the other hand, `PGN-BERT-BERT` refer to the setting where the beams are further re-ranked with the BERT-based re-ranker described previously.

3.7 Discussion

T5 not only outperforms all previous works on both datasets, but also BART and PGN that we experiment with. It is important to handle input tokenisation of prefixes through sentinel IDs for T5, or else the model produces zero accuracy. In the case of PGN, we also performed a test without input KG embeddings, where we saw a 20% point drop in accuracy. In spite of the addition of KG embeddings, it fails to get an edge over T5, which operates solely based on entity and relation labels. It can hence be concluded that the large amount of pre-training for T5 outranks the KG embedding advantage.

In the case of BART versus T5, we observed that the copy actions of BART were not as clean as T5. For example, in LC-QuAD 2.0, BART produced an F1 of 0.3 for questions that require token copy to the output, while T5 produced an F1 of 0.8. We believe this is due to the fact that instead of using sentinel IDs, which are inherently a part of the T5 model from its pre-training objective, we had to resort to adding special tokens to the `BartTokenizer`, which does not produce the same copying performance.

`T5-small` and `T5-base` produce similar performance, which suggests that the extra number of parameters in the base model remains unused for our given task. For LC-QuAD 2.0, none of the previous works has the ability to copy tokens from input to the output, so our approaches outperform them by far. In terms of PGN performance, on using the BERT re-ranker, we see gains in the range of 10-20 % points, which shows that PGN does a poor job of producing the right query towards at the top. When comparing

[5] https://kgqa.github.io/leaderboard/
[6] https://databus.dbpedia.org/dbpedia/wikidata/debug/2020.07.01
[7] https://dumps.wikimedia.org/wikidatawiki/entities/

T5-small performance with PGN, although they have similar parameter count, the pre-training on 750 GB of corpus seems to be the crucial factor in T5-small's favour.

3.8 Error Analysis

We randomly sampled 100 cases of erroneous outputs for PGN, T5-Small and BART, as shown in Table 3.2. The most important difference in the errors produced by T5/BART versus PGN is what we call "copy morphing". T5/BART produce their output from an open ended sub-word-based vocabulary, and hence sometimes corrupt the item being copied. At times, they produce wrong entity IDs, e.g., in the case of DBpedia, Barack-Obama instead of Barack_Obama, and in some other cases, they hallucinate entity IDs that are not part of the linked entities or relations in input. We also encountered cases of morphing relation IDs to semantically similar words, like notableWork changing to notabilityWork, or unexpected capitalisation, e.g., Artist instead of artist. Although PGN's overall accuracy is lower, they do not exhibit this issue, since the copying takes place from an expanded vocabulary which is limited, consisting of SPARQL tokens and the input tokens. We also found that copy morphing in T5 takes place only for DBpedia where the entity and relation IDs are similar to the dictionary labels. For Wikidata, where the IDs are numerical, morphing does not occur.

We categorise the other kind of errors under the following heads: Triple Flip, e.g., <o p s> instead of <s p o>, Wrong Variables, e.g., ?var0 instead of ?var1, Wrong Intent, e.g., ASK instead of SELECT, Copy Errors, where the wrong token is copied, and Syntax Errors, where the generated query is malformed and an invalid SPARQL query. We observe that BART has poor performance in copying, while other models have most errors in triple flips and wrong variables.

	PGN		BART		T5-Small	
	LCQ1	LCQ2	LCQ1	LCQ2	LCQ1	LCQ2
Triple Flip	56	54	22	30	66	60
Wrong Var	78	63	18	-	36	8
Wrong Intent	22	15	-	10	4	26
Copy Error	-	14	22	40	-	18
Copy Morph	-	-	60	-	26	-
Syntax Errors	-	-	16	40	-	-

Table 3.2: Error breakdown for randomly sampled 100 errors

3.9 Conclusion and Future Work

In this work, we establish new baselines using PLMs for the KG semantic parsing task for two datasets. Our results surpass the results of earlier baselines and lay grounds for future research. We make use of readily available and popular models, with minimal changes to input vectorisation and tokenisation, to report state of the art results on SPARQL semantic parsing and query building tasks over DBpedia and Wikidata. We show that these methods are flexible and able to handle a variety of questions with no fixation on templates or query graphs. The direction of future semantic parsing

work for SPARQL should lay more emphasis on pre-training, since currently it is more focused on producing custom model architectures for the tasks.

As future work, we would like to explore the ability of these models in disambiguation tasks, where the input consists of entity and relation candidates, instead of linked entities and relations.

3.10 Acknowledgments

This research was partially funded by the German Federal Ministry of Education and Research (BMBF) as part of the INSTANT project, ID 02L18A111.

4

GETT-QA: Graph Embedding based T2T Transformer for Knowledge Graph Question Answering

4.1 Bibliographic Information

Debayan Banerjee, Pranav Ajit Nair, Ricardo Usbeck, Chris Biemann. 2023. GETT-QA: Graph Embedding Based T2T Transformer for Knowledge Graph Question Answering. In Proceedings of The Semantic Web: 20th Extended Semantic Web Conference. Hersonissos, Crete, Greece. Pages 279–297. https://doi.org/10.1007/978-3-031-33455-9_17

4.2 Abstract

In this work, we present an end-to-end Knowledge Graph Question Answering (KGQA) system named GETT-QA. GETT-QA uses T5, a popular text-to-text pre-trained language model. The model takes a question in natural language as input and produces a simpler form of the intended SPARQL query. In the simpler form, the model does not directly produce entity and relation IDs. Instead, it produces corresponding entity and relation labels. The labels are grounded to KG entity and relation IDs in a subsequent step. To further improve the results, we instruct the model to produce a truncated version of the KG embedding for each entity. The truncated KG embedding enables a finer search for disambiguation purposes. We find that T5 is able to learn the truncated KG embeddings without any change of loss function, improving KGQA performance. As a result, we report strong results for LC-QuAD 2.0 and SimpleQuestions-Wikidata datasets on end-to-end KGQA over Wikidata.

4.3 Introduction

A Knowledge Graph (KG) is an information store where data is stored in the form of node-edge-node triples. Nodes represent entities and edges represent relationships between these entities. The aim of KGQA (Lan, G He, et al., 2021) is to produce answers from this KG given an input question in natural language, e.g., `Who is the father of Barack Obama ?`. Usually, the first step in KGQA is to perform Entity Linking (EL) where mention spans, e.g., `Barack Obama` representing the name of a person, place, etc., are linked to a KG node. The subsequent step is Relation Linking (RL), where the relationship of the entity to the potential answer in the KG is extracted, e.g., `father of`. Some KGQA systems attempt to fetch the answer based on the results of just the two steps above, which typically ends up being another entity (node) in the graph. However, for more complex questions, such as count queries or min/max aggregate queries (e.g.: `How many rivers are there in India`?) the answer does not lie in a node or edge in the graph, but instead, a formal query must be generated as a final step. To this end, semantic parsing is relevant to the problem of KGQA. Thus, our focus in this work is to generate a final SPARQL query that can be executed on the KG.

SPARQL is a popular graph query language for querying KGs. A sample SPARQL query for the running example over the Wikidata KG looks like the following:

```
SELECT ?o WHERE { wd:Q76 wdt:P22 ?o }
```

In the query above, `wd:Q76` stands for Barack Obama, while `wdt:P22` stands for the relation `father`. The `?o` variable represents the answer from the KG.

Recent works employ text-to-text (T2T) pre-trained language models (PLMs) for generating logical queries, e.g. SPARQL, from natural language questions. If the correct entity and relation IDs are already specified in the input, the accuracy of T2T models is high (Banerjee, Nair, Kaur, et al., 2022). However, the absence of linked entity and relation IDs in the input presents a significant challenge to such models. PLMs are adept at generating linguistic tokens from within their weights. Yet, it is an entirely different proposition to query the KG and ground the entity and relations to specific IDs, as the variability of language creates impressive richness at generation while at the same time hampers the alignment to pre-defined KG items.

In this work, we demonstrate a novel method by which a T2T PLM, namely T5 (Raffel et al., 2020), not only generates SPARQL queries, but also generates truncated KG embeddings, which aid in the subsequent process of grounding entities to the correct node in the KG. Our method produces strong results for end-to-end Question Answering on the LC-QuAD 2.0 and SimpleQuestions-Wikidata datasets over Wikidata KG. All code and data will be made available [1].

4.4 Related Work

Early KGQA systems could be divided on the basis of whether they can handle simple (Yani and Krisnadhi, 2021) or complex questions (Lan, G He, et al., 2021). In a simple

[1] https://github.com/debayan/gett-qa

question, a node-edge-node triple is a sole basis on which a question is formed, whereas in a complex question there may be more than one such triple involved. Moreover, certain KGQA systems are built specifically to handle a certain class of questions better, e.g. temporal questions (Jia et al., 2021).

Another way of categorising KGQA systems is whether they form a formal query (Christmann et al., 2019; Bhutani et al., 2019; Diefenbach, Both, et al., 2020; Shen et al., 2019; Tanon et al., 2018) versus whether they use graph search based methods without producing an explicit query (Christmann et al., 2019; Vakulenko et al., 2019; Huang et al., 2019; Saxena et al., 2020; H Sun, Bedrax-Weiss, et al., 2019; H Sun, Dhingra, et al., 2018; Ravishankar et al., 2021; Neelam et al., 2022).

Some KGQA systems work in a hybrid mode and can query from both KG and text-based sources. PullNet (H Sun, Bedrax-Weiss, et al., 2019) and Graftnet (H Sun, Dhingra, et al., 2018) both use Relational-Graph Convolution Networks (Schlichtkrull et al., 2018) to handle complex questions. UNIK-QA (Oguz et al., 2022) verbalises all structured input from KG, tables and lists into sentences and adds them to a text corpora and proceeds to perform QA over this augmented text corpora using deep passage retrieval techniques. UNIQORN (Pramanik et al., 2021) builds a context graph on-the-fly and retrieves question relevant pieces of evidence from KG and text corpora, using fine-tuned BERT models. They use group Steiner trees to identify the best answer in the context graph. We use the results of the KG components of these hybrid systems in our evaluation in Table 3.1, as reported by UNIQORN.

Platypus (Tanon et al., 2018) and QAnswer (Diefenbach, Both, et al., 2020) are two recent KGQA systems that work on Wikidata. Both of them use templates and ranking mechanisms to find the best query. We make no use of templates in our method since this inherently limits the flexibility of a system on unseen templates.

ElneuQA-ConvS2S (Diomedi and Hogan, 2021) operates in a similar fashion to us, where they use a Neural Machine Translation (NMT) model to generate SPARQL queries with empty placeholders, while an entity linking and sequence labeling model fills the slots. In our case we also make use of NMT capabilities of T5 to generate a skeleton SPARQL query, however, we do not generate empty slots, and instead, generate entity and relation labels to be grounded later.

For simple questions, KEQA (Huang et al., 2019) targets at jointly recovering the question's head entity, predicate, and tail entity representations in the KG embedding spaces and then forming a query to retrieve the answer from a KG. Text2Graph (Chekalina et al., 2022) uses KEQA as a base, and improves on the embedding learning model by utilising CP tensor decomposition (Hitchcock, 1927). We include both these systems in our evaluation Table 4.4.

SGPT (Rony et al., 2022) and STAG (Ravishankar et al., 2021) both use generative methods for forming the query using pre-trained language models, which is similar to what we do, however, neither of them generate the entity or relation label, or the embeddings. Instead STAG uses an external entity linking step, while SGPT attempts to generate entity and relation IDs directly. However such a method does not work well because for a KG like Wikidata, the IDs do not follow a hierarchical pattern, and hence the model is not able to predict an ID that it has not seen in training earlier.

One of our key ideas is to enable a PLM to learn KG Embeddings. There have been some recent efforts in the same direction such as KEPLER (Xiaozhi Wang et al., 2021), K-BERT (W Liu et al., 2020), KI-BERT (Faldu et al., 2021), CoLAKE (T Sun et al., 2020), BERT-MK (B He et al., 2020) and JAKET (D Yu et al., 2022). These systems either try

to inject KG embeddings into the input or intermediate layers of the model, or they try to augment the text corpora by including verbalised forms of the triple structural information. On the other hand, we ask the model to print the embeddings as output. This is a fundamentally different approach from what has been tried so far.
A related, yet different class of systems is that of conversational QA.
LASAGNE (Kacupaj et al., 2021) and CARTON (Plepi et al., 2021) are two notable systems in this category. They evaluate on the CSQA dataset (Saha et al., 2018), which is a conversational dataset answerable over a KG. In our case, we address only single sentence-long questions. The conversations in CSQA are arranged in sequence of turns of questions and answers. For the semantic parsing of logical forms, both LASAGNE and CARTON use a pre-defined grammar, while our approach is free of templated grammar rules. Both LASAGNE and CARTON use a Transformer architecture to generate base logical forms, however, LASAGNE uses, a Graph Attention Network (Velickovic et al., 2018) to resolve entities and relations while CARTON uses stacked pointer networks.

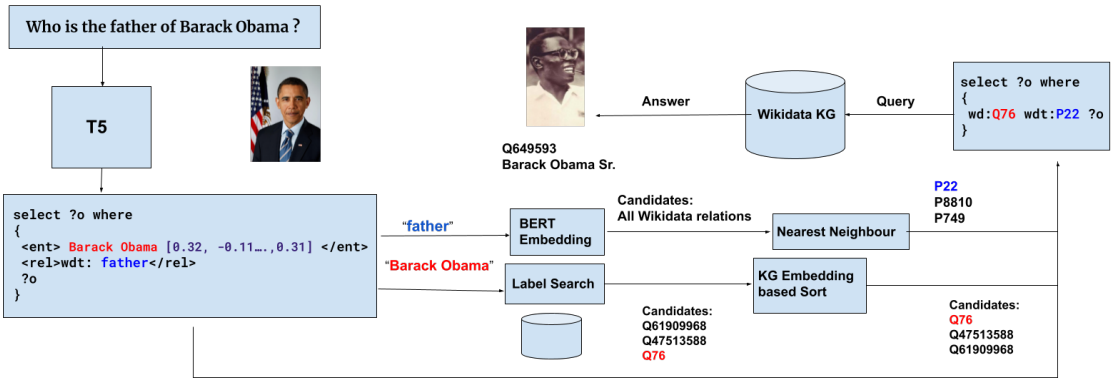


Figure 4.1: Architecture of GETT-QA: T5 generates a skeleton SPARQL query with entity and relation labels and a truncated KG embedding string. The entity labels are grounded using label based search and embedding based sorting, while the relations are grounded using BERT embedding based sorting. The final query is executed against a KG to fetch the answer.

4.5 Method

As shown in Figure 4.1, our system consists of five major steps:

- T5 generates a skeleton SPARQL query from input natural language question.
- The entity labels and truncated KG embeddings are extracted. A label search is performed to fetch entity candidates.
- The entity candidates are re-ranked using an embedding similarity metric.
- In parallel, the relation label is extracted and matched against Wikidata relations based on BERT embeddings.
- The final query is grounded to KG nodes and executed to receive the answer.

4.5.1 Truncated KG Embeddings

We teach T5 to generate truncated vector strings of KG embeddings. We use TransE (Bordes, Usunier, Garcia-Duran, et al., 2013) embeddings for Wikidata entities that were provided by Pytorch-BigGraph[2] (Lerer et al., 2019). These are 200-dimensional vectors of floats. The truncated KG embeddings we use are a shorter version of the same embeddings. For most of our experiments, we use the first 10 dimensions of these embeddings, and further reduce the precision of the floats to 3 digits after the decimal. We do so since T5 is expected to produce these truncated KG embeddings while still in the text-to-text mode. In other words, T5 produces these vectors of floats considering them as a string. We use truncated KG embeddings instead of original embeddings to reduce the decoding load of T5. Our aim is not to learn the entire embedding space. Instead, we want to learn identifiers that can aid the entity disambiguation phase. We produce these truncated KG embeddings only for entities, not for relations.

4.5.2 Intuition

The initial idea behind our approach is to allow T5 to use its significant linguistic capability derived from pre-training over a large corpus for the task of semantic parsing. As a result, we use T5 to produce SPARQL tokens and entity and relation labels.

At first glance, it may appear that the production of entity labels is sufficient for grounding to KG entity nodes. However, in most KGs, several entities share the same labels. For example in Wikidata KG, the entity IDs `Q76` and `Q47513588` both share the label `Barack Obama`. In reality, `Q76` represents the President while `Q47513588` is the entity ID for a painting of the President. As a result of such collision of labels, a further step called *disambiguation* is required.

The next idea is to not just rely on T5's linguistic abilities but also to try and teach the model how to generate identifiers for the entities, which can aid the grounding and disambiguation process in a subsequent step. One way could be to generate the Wikidata IDs directly. However, the IDs do not correspond in any hierarchical way to the underlying entities. For example, while `Q76` is `Barack Obama`, `Q77` is `Uruguay`. Although the IDs are close to each other, the categories are completely different. Models cannot be expected to produce accurate IDs of this nature, especially on unseen input questions. As a result, we consider other schemes of entity identifiers that exhibit some hierarchical properties.

It turns out KG embeddings fulfill these requirements handsomely, and hence we decide to use truncated KG embeddings as the "soft" identifier of our choice. Another possibility would be to generate entity descriptions instead of truncated KG embeddings, however, roughly 50% of entities in Wikidata do not have corresponding descriptions (eg: Q67395985[3]), hence we focus on generating truncated KG embeddings instead.

While the production of such truncated embeddings may also aid the grounding of relations, we do not attempt this, since Wikidata only contains a few thousand relations, while the number of entities run into several millions. For the grounding of relations we use simpler text embedding based methods, as described later in Section 4.5.7.

[2] https://github.com/facebookresearch/PyTorch-BigGraph
[3] https://www.wikidata.org/wiki/Q67395985

4.5.3 Models

T5 (Raffel et al., 2020), or text-to-text transfer transformer, is a transformer (Vaswani et al., 2017) based encoder-decoder architecture that uses a text-to-text approach. Every task, including translation, question answering, and classification, is cast as feeding the model text as input and training it to generate some target text. This allows using the same model, loss function, hyper-parameters, etc., across a diverse set of tasks. We experiment with fine-tuning two variants of the T5 model: `T5-Small` with 60 million parameters and `T5-Base` with 220 million parameters. For the GETT-QA system results reported in Tables 3.1 and 4.4 we use the `T5-Base` based model, whereas in the analysis Section 4.9.2 we present a comparative study against `T5-Small`.

4.5.4 Skeleton SPARQL

As shown in Figure 4.1, the first step of our KGQA system is to generate a skeleton SPARQL query from the given natural language question. The skeleton query consists of SPARQL tokens such as `SELECT, WHERE, {}`, entity and relation labels, and truncated KG embeddings, which are an array of `floats`. Some additional tokens are used to surround the entity and relation labels, such as `<ent>, </ent>,<rel>,</rel>` so that in a later step their extraction can be performed using regular-expression operations. The extraction of the labels and the truncated KG embedding are essential for the subsequent grounding step. Notably, entity and relation IDs are not a part of a skeleton SPARQL query.

During training via fine-tuning, pairs of questions and skeleton SPARQL queries are presented to T5. For this purpose, we pre-process the original dataset, which contains gold SPARQL queries for each question. The SPARQL query is converted to a skeleton SPARQL query by replacing the entity and relation IDs with their gold labels while appending the entity labels with a truncated KG embedding. Hence, the following gold SPARQL query:

```
select ?o where {  wd:Q76 wdt:P22 ?o }
```

is converted to:

```
select ?o where
{
 <ent>Barack Obama [...] </ent>
 <rel>father</rel>
 ?o
}
```

for the purposes of training T5, where the truncated KG embedding is represented by `[...]` .

4.5.5 Entity Candidates

During inference, when T5 generates a skeleton query, all entity and relation labels, as well as truncated KG embeddings, are extracted using regular expressions. For the entity labels, a BM-25-based (Robertson and Walker, 1994) label search is performed on a

database of all Wikidata entity labels, out of which top-k candidates are retrieved per entity label. For this text search we use the Elasticsearch database[4]. For our experiments, we fix k at 100.

4.5.6 Entity Candidates Re-ranking and Ordering

The top-3 entity candidates based on label matching are retained. For the next 3 candidates, we resort to truncated KG embedding-based sorting. For each item in the list of 100 entity candidates fetched, we also fetch their gold KG embeddings, and convert them to truncated KG embeddings. For the truncated KG embedding generated by T5, we compute its dot product against the gold truncated KG embeddings fetched and re-rank them in descending order. The dot product is used as a comparator because this was the same function that was used during the production of the TransE embeddings. From this re-ranked list based on truncated KG embedding similarity, top-3 candidates are retained.

We append these `top-3` truncated KG embedding-sorted candidates to the `top-3` label-sorted candidates, and proceed to the subsequent steps with a list of 6 candidate entities for each entity label.

4.5.7 Relation Candidates

We generate no truncated KG embeddings for relation IDs, as their numbers are orders of magnitudes smaller when compared to entities in Wikidata. From the relation labels generated by T5, we compute their BERT embeddings and compute the cosine similarity against the BERT embeddings of all Wikidata properties. The list of properties is sorted based on this similarity score, and the top-3 matches are considered for the subsequent steps.

4.5.8 Candidate Combinations

For a generated query, each entity label and each relation label has 6 and 3 candidates each, respectively. We preserve the serial order of the entities and relations as produced in the query, and generate all possible combinations of the entities and relations, which generates several queries of the same structure but different entity and relation IDs. For example, if the query contains just one entity and one relation, the number of possible SPARQL queries generated would be $6 \times 3 = 18$. We execute each query on the KG in sorted order of entity and relation IDs received in previous steps. We stop when the KG returns a non-empty response. This response is considered the output of our KGQA system. We consider the top 3 beams produced by T5 decoder as probable queries. The first beam producing a valid response from the KG is considered the output of our KGQA system.

4.6 Dataset

We evaluate our approach on the LC-QuAD 2.0 (Dubey, Banerjee, Abdelkawi, et al., 2019) dataset, which consists of approximately 30,000 questions based on the Wikidata KG. Each question contains the corresponding SPARQL query as gold annotation. The

[4] https://www.elastic.co/

Question	SPARQL

Tell me the female beauty pageant that operates in all countries and contains the word model in it's name?	`SELECT DISTINCT ?sbj ?sbj_label`
	`WHERE {`
	`?sbj wdt:P31 wd:Q58863414 .`
	`?sbj wdt:P2541 wd:Q62900839 .`
	`?sbj rdfs:label ?sbj_label .`
	`FILTER(CONTAINS(lcase(?sbj_label), "model")) .`
	`FILTER (lang(?sbj_label) = "en")`
	`}`
	`LIMIT 25`

Table 4.1: Sample question from LC-QuAD 2.0

Question	SPARQL
What type of music does David Ruffin play ?	`SELECT ?x WHERE { wd:Q1176417 wdt:P136 ?x }`

Table 4.2: Sample question from SimpleQuestions-Wikidata

dataset consists of a wide variety of questions, such as simple, complex, multi-hop, count, min/max, dual and boolean types. This dataset also uses the recently introduced hyper-relational (Galkin et al., 2020) structure of the Wikidata KG.
Additionally, we evaluate our approach on the SimpleQuestions-Wikidata (Diefenbach, Tanon, et al., 2017)[5] dataset, which consists of 34,374 train questions and 9,961 test questions. This dataset is derived from the original SimpleQuestions dataset (Bordes, Usunier, Chopra, et al., 2015), which was later aligned with the Wikidata KG. A sample question from each dataset can be seen in Tables 4.1 and 4.2.

4.7 Evaluation

In Table 3.1, the results for UNIQORN, QAnswer, UNIK-QA, Pullnet, and Platypus are taken from UNIQORN (Pramanik et al., 2021). UNIQORN uses a test split of 4,921 questions from the original LC-QuAD 2.0 test set of 6,046 questions for all the systems. We evaluate our approach on the same split as UNIQORN. Despite our best efforts we were unable to acquire the precise KG snapshot that UNIQORN used for evaluation. UNIQORN used a Wikidata dump dated 20 April 2021, which is no longer available either in the official Wikidata repository[6], or with the authors of UNIQORN. As a result, we ran the 4,921 questions against the NLIWOD[7] Wikidata dump[8],[9], which

[5] https://github.com/askplatypus/wikidata-simplequestions
[6] https://dumps.wikimedia.org/wikidatawiki/entities/
[7] https://www.nliwod.org/challenge
[8] https://hub.docker.com/r/debayanin/hdt-query-service
[9] https://skynet.coypu.org/#/dataset/wikidata/query

	P@1
UNIK-QA	0.005
Pullnet	0.011
Platypus	0.036
QAnswer	0.308
UNIQORN	0.331
GETT-QA without truncated embeddings	0.327 ± 0.002
GETT-QA (with truncated embeddings)	**0.403** ± 0.0

Table 4.3: Results on LC-QuAD 2.0

	F1
KEQA	0.405
Text2Graph	0.618
GETT-QA without truncated embeddings	0.752 ± 0.004
GETT-QA (with truncated embeddings)	**0.761** ± 0.002

Table 4.4: Results on SimpleQuestions-Wikidata

is hosted on the docker hub for easy deployment, and also hosted as an API by the Universität Hamburg's SEMS group.

In Table 4.4 for SimpleQuestions-Wikidata, results for KEQA and Text2Graph are taken from MEKER (Chekalina et al., 2022). They evaluate both systems on a smaller split of the SimpleQuestions-Wikidata test set. This subset contains those questions which are valid on a custom Wikidata version they call Wiki4M. We were provided the KG by the authors of MEKER and we evaluated our system on the same.

In Table 3.1, we report macro Precision@1 based on UNIQORN's reporting preference. In Table 4.4 we report macro F1 in line with MEKER. To compute metrics, we take the gold SPARQL and predicted SPARQL query and query the KG with both. We compare the results from the KG to compute true positives, false positives, and false negatives (TP, FP, FN).

4.8 Results

In Table 4.3, the bottom two rows contain the results of our system in two different settings for LC-QuAD 2.0. In the first case, our KGQA system uses the top-6 entity candidates based on label match, without the use of truncated KG embeddings for re-ranking. In the second case, we keep the top-3 entity candidates based on label match and append to it the top-3 candidates based on truncated KG embedding match. This is the same setting as described in Subsection 4.5.6. The relation candidates in both cases remain top-3 as described in Subsection 4.5.7.

The key finding in Table 4.3 is that when we compare the last two rows, our system performs better with an absolute gain of approximately 8% when truncated KG embeddings are used.

In Table 4.4, for SimpleQuestions-Wikidata our method without truncated KG embeddgings already outperforms the nearest competitor by an absolute margin of 13%. This demonstrates the natural ability of T5 to predict the correct entity and

relation labels given a question. Since the query structure of all the questions in this dataset is the same, no challenge is posed to T5 in trying to learn the query itself. It is noteworthy however, that after the inclusion of truncated KG embedding re-ranking, the performance remains similar. An insignificant margin of absolute improvement of 0.9% is seen. To investigate this gap when compared to the 8% improvement of LC-QuAD 2.0, we delve further into the nature of the two datasets and run some analysis. We find that in the case of LC-QuAD 2.0, the correct entity is found in the top-1 position of the candidate list based on text match 60% of the time, whereas in the case of SimpleQuestions-Wikidata, this number is significantly higher at 82%. This is so because the questions in SimpleQuestions-Wikidata contain the entity names in almost the exact form as their gold entity annotations, whereas in LC-QuAD 2.0, several entity labels are modified, misspelled or shortened by the human annotators. Hence, in the case of SimpleQuestions-Wikidata, label-only matching is in most cases sufficient, whereas in LC-QuAD 2.0, truncated KG embedding-based disambiguation holds greater importance.
Additionally, as mentioned in Section 4.4, STAG (Ravishankar et al., 2021) and ElNeuQA-ConvS2S (Diomedi and Hogan, 2021) are comparable generative KGQA systems. For STAG, no code, data or KG versions have been made public, while for ElNeuKGQA we were unable to run their code[10] as no instructions on how to run the code exists. For STAG on SimpleQuestions (Bordes, Usunier, Chopra, et al., 2015), on a test split of 2280 questions they report F1 61.0 while we report F1 78.1 on a larger test split of 9961 questions. Lastly, ElNeuQA-ConvS2S reports an F1 of 12.9 on WikidataQA while we report 17.8. WikidataQA is a 100 question test-subset created by the authors of ElNeuQA-ConvS2S. On LC-QuAD 2.0, they report F1 of 26.9 while we report 40.3.

4.8.1 Limitations

Although GETT-QA performs the best in Tables 3.1 and 4.4, we do not claim state-of-the-art results on the respective datasets. This is due to a variety of reasons: as mentioned in Subsection 4.7, we could not procure the precise Wikidata KG version as the competing systems for LC-QuAD 2.0. In the case of LC-QuAD 2.0 and SimpleQuestions-Wikidata, evaluation was performed on a truncated subset of the original test split of this dataset. As a result we can not claim that we have the best results on the entire dataset. Additionally, we could not find the code for, or run majority of systems we evaluated against, and hence resorted to using the results as reported by them.

4.9 Analysis

4.9.1 Error Analysis

In an attempt to find the common source of errors in LC-QuAD 2.0 we find that by far the largest cause of incorrect answers is the improper grounding of entities and relations to nodes in the KG. More than 95% of questions where the correct entities and relations were in the top-6 and top-3 candidates respectively, the right answer was eventually produced by the KG. Unfortunately, only in 41% of the questions, the correct entities and relations were found within the top-k candidates. This suggests that greater

[10] https://github.com/thesemanticwebhero/ElNeuKGQA

focus in the area of entity and relation linking will produce better results. One may also increase the size of k, at the cost of increased run time.

Less than 1% of the queries generated had incorrect truncated KG embedding length (e.g.: 11 instead of 10) however these were handled in code appropriately. Less than 1% of queries generated were improper SPARQL, where a critical keyword was missing rendering the query syntactically incorrect. This suggests that T5 learns how to generate valid SPARQL queries to a large extent. This is consistent with the findings of Banerjee, Nair, Kaur, et al. (2022) where T5 crosses 90% accuracy when provided with grounded entity and relation IDs with their labels.

To explore the issue of lack of correct entities and relations in candidate lists, we observe that while 60% of questions contain the correct entity ID in the top-100 label-based search candidates, by the time we reduce this list to top-6, only 49% questions remain with the correct entity ID in the candidates list. On the other hand, for relation candidates, 45% of questions contain the correct relation IDs in the top-3 relation candidates.

When looking at the category of questions that return incorrect KG responses, irrespective of entity and relation grounding, we find that `COUNT` queries are the most common. This happens due to a quirk in SPARQL format. If a `COUNT` query is built around a family of triples that do not exist in the KG, the KG responds with `count = 0`, instead of producing a `NULL` value or an `ERROR`. This means that the very first query to be executed on the KG with the correct `COUNT` SPARQL syntax will return a valid response, even if it is `count = 0` and we no longer explore subsequently ranked queries. In the case of SimpleQuestions-Wikidata, all errors are either due to incorrect entity or relation linking. The model produced an accuracy of 70% for entity linking and 94% for relation linking.

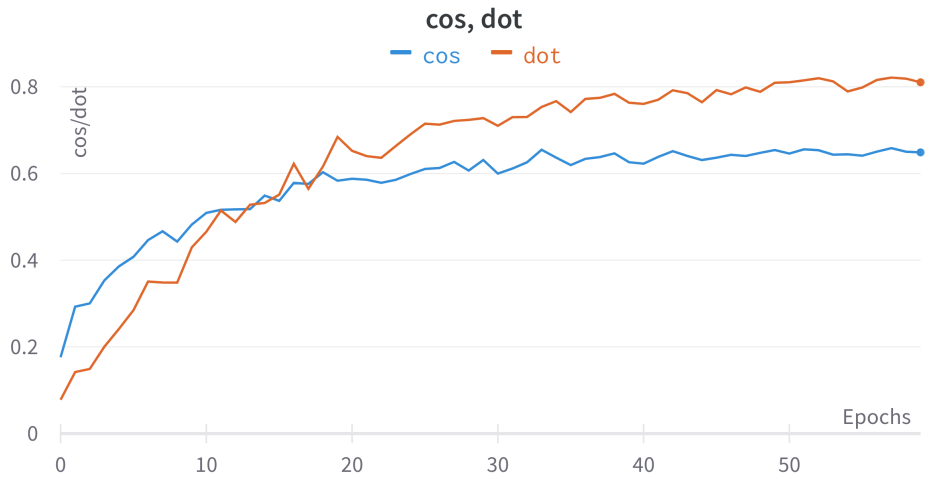


Figure 4.2: Cosine and Dot Product based similarities of truncated KG embeddings

4.9.2 Truncated KG embedding Learning

We discover that T5 is able to produce a vector of floats while still in the text-to-text mode of decoding. For this functionality, no change of loss function or decoding scheme is made in our experiments. It effectively learns a simplified embedding space, but with

certain limitations. To further explore this ability of T5, we performed some additional experiments on the 200 questions dev set of LC-QuAD 2.0 with `T5-Small`.

In Figure 4.2, we compare how the model learns the embedding space with each epoch of training. The TransE embeddings have an angular component and a magnitude component. Since the dot operation was used to train the original embeddings, the magnitude of each embedding may be greater than 1. It appears in Figure 4.2 that around step 20 the angular component of the embeddings has been learned by T5 to the best of its abilities, and it proceeds to learn the magnitude component (denoted by the orange line) further.

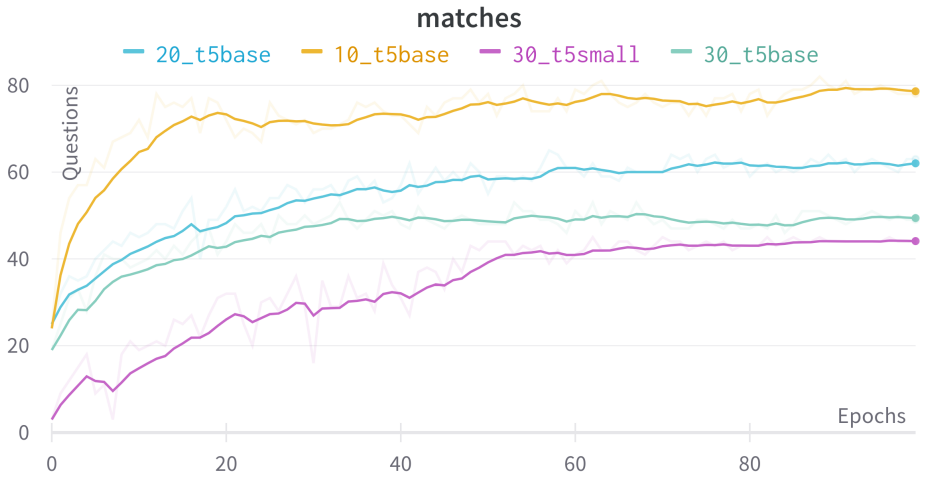
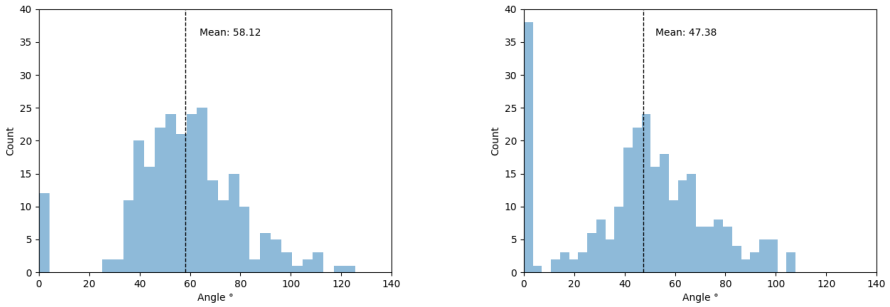


Figure 4.3: Dev Set matches for varying truncated KG embedding lengths

In Figure 4.3 we chart the LC-QuAD 2.0 dev set performance of `T5-Small` and `T5-Base` in varying truncated KG embedding lengths. For `T5-Small` (pink line) we set the truncated KG embedding length at 30, so this should only directly be compared against `T5-Base` with truncated KG embedding length 30 (green line). In the matches metric, which only looks at the keywords and labels produced in the skeleton SPARQL query (truncated KG embeddings have been removed from the generated query), the two reach similar performance. This suggests that a larger number of parameters helps learn the embedding space better, but for the textual component the extra number of parameters of `T5-Base` remain unused.

With a truncated KG embedding length of 10 (yellow line), we see the best label match accuracy and hence we persist with this family of models for reporting results in Table 3.1.

In Figure 4.4, we plot the distribution of the angular difference between the gold and predicted truncated embeddings on the LC-QuAD 2.0 dev set. The model seems to learn the embedding space in two distinctly different manners: firstly, for several entities it is able to print the exact embedding, with an angular difference of 0°. Secondly, for the entities which it is unable to learn the embedding of exactly, it produces a more familiar distribution, where the mean shifts every few epochs, reducing the angular difference. This suggests that the model is learning the embedding space effectively.



(a) 10 epochs (b) 40 epochs

Figure 4.4: Distribution of angular difference between gold and predicted truncated KG embeddings on LC-QuAD 2.0 dev set. The mean angular difference can be seen reducing as the epochs progress, suggesting that the model is learning the embedding space.

	F1
3 LS + 3 TS	0.365
3 TS + 3 LS	0.331
3 LS + 0 TS	0.289
0 LS + 3 TS	0.236
6 LS + 0 TS	0.319
0 LS + 6 TS	0.256

Table 4.5: Effects of ordering entity candidates differently. LS = Label sorted, TS = Truncated KG embedding sorted

4.9.3 Candidate Ordering

As mentioned in Subsection 4.5.6, our results reported in Table 3.1 come from a configuration of our system where the entity candidates are layered in two parts: the first three candidates are sorted based on label match, while the bottom three candidates are sorted based on the truncated KG embedding dot product similarity. It is observed in Table 4.5 that the ordering of these two categories affects the eventual accuracy strongly. We take 200 questions at random from the LC-QuAD 2.0 test set and perform experiments to ascertain how the ordering of candidates affects accuracy. In the first row of the table, three entity candidates based on label sorting are followed by three candidates of truncated KG embedding sorting, while the next row of the table shows the result when we keep truncated KG embedding-sorted candidates above label-sorted candidates. The results show that keeping truncated KG embedding-sorted candidates at the top reduces accuracy, and hence, label-based matching for entities remains a stronger mode of fetching correct candidates. This is no surprise, since not all labels have multiple entity candidates requiring disambiguation. Kindly note that the accuracy drops because an earlier query formed due to candidate combinations (as explained in Subsection 4.5.8) returns a non-empty result from the KG and this response turns out to be incorrect.

In the next two rows, we see how excluding either label-sorted candidates, or truncated KG embedding-sorted candidates entirely affects accuracy. Once more we see that label-sorted candidates still perform better when used in isolation. However,

the crux of the table is the first row, i.e., when both the categories are appropriately sequenced and used in tandem, the accuracy is best.

In the bottom two rows, we see the effect of changing the number of candidates. It is no surprise that increasing k from 3 to 6 increases accuracy since more correct entities are included in the list. However, some systems also perform worse in such settings as the noise may increase adding to the disambiguation load. In our system, this does not seem to be the case.

Increasing the value of k further imposes a large cost on the run time of the system affecting the user experience adversely. Since the candidate combination step has exponential complexity, which depends on the number of entities and relations in a query, we need to keep the number of candidates in check. Too many candidates will produce too many SPARQL queries and the user must wait for all of them to be executed on the KG till one of them responds validly.

In our choice of setting k=6 for entities and k=3 for relations, we observe that on the test set of LC-QuAD 2.0, our system has an average response time of 1.2 seconds per question, which from a user experience perspective seems like an acceptable response time.

4.10 Hyperparameters and Hardware

For the evaluation of LC-QuAD 2.0 in Table 3.1, we fine-tune our models for 50 epochs with a learning rate of 1e-04 with the Adam optimizer (Kingma and Ba, 2015). For SimpleQuestions in Table 4.4 we fine-tune for 25 epochs, roughly half of LC-QuAD 2.0, since the train set is roughly twice as large as LC-QuAD 2.0. We use a batch size of 8. During this phase we had access to NVIDIA GeForce RTX 2080 Ti/1080 Ti graphics cards with approximately 11GB of video memory. We do not fix a seed during training, and train and infer three times. We report mean and standard deviation for the three runs in the respective tables.

For the analysis in Section 4.9, we fine-tune our models for 100 epochs with a learning rate of 1e-05 with the Adam Optimizer (Kingma and Ba, 2015) and we use a batch size of 20. During this phase we had access to larger GPUs, namely NVIDIA RTX A6000 with 48GB of memory and RTX A5000 with 24GB memory.

4.11 Conclusion and Future Work

In this work, we presented a novel KGQA pipeline called GETT-QA. We use no external entity or relation linking tool, and still achieve strong results on LC-QuAD 2.0 and SimpleQuestions-Wikidata datasets. Additionally, we discover the ability of T5 to learn KG embeddings. We demonstrate that in certain situations this ability helps in better question answering performance.

In future work, we will explore the ability of T5 in generating similar truncated KG embedding based queries with modified loss functions and a customised architecture towards the penultimate layers of the models, so that embeddings can be generated with more standard loss functions meant specifically for learning embeddings. Additionally, suitable identifiers other than embeddings can also be explored, for example, text description based identifiers.

4.12 Acknowledgements

This research was supported by grants from NVIDIA and utilized NVIDIA 2 x RTX A5000 24GB. Furthermore, we acknowledge the financial support from the Federal Ministry for Economic Affairs and Energy of Germany in the project CoyPu (project number 01MK21007[G]) and the German Research Foundation in the project NFDI4DS (project number 460234259). This research is additonally funded by the “Idea and Venture Fund“ research grant by Universität Hamburg, which is part of the Excellence Strategy of the Federal and State Governments.

5

The Role of Output Vocabulary in T2T LMs for SPARQL Semantic Parsing

5.1 Bibliographic Information

Debayan Banerjee, Pranav Ajit Nair, Ricardo Usbeck, and Chris Biemann. 2023. The Role of Output Vocabulary in T2T LMs for SPARQL Semantic Parsing. In Findings of the Association for Computational Linguistics. Toronto, Canada. Pages 12219-12228. https://doi.org/10.18653/v1/2023.findings-acl.774

5.2 Abstract

In this work, we analyse the role of output vocabulary for text-to-text (T2T) models on the task of SPARQL semantic parsing. We perform experiments within the the context of knowledge graph question answering (KGQA), where the task is to convert questions in natural language to the SPARQL query language. We observe that the query vocabulary is distinct from human vocabulary. Language Models (LMs) are predominantly trained for human language tasks, and hence, if the query vocabulary is replaced with a vocabulary more attuned to the LM tokenizer, the performance of models may improve. We carry out carefully selected vocabulary substitutions on the queries and find absolute gains in the range of 17% on the GrailQA dataset.

5.3 Introduction

Knowledge Graph Question Answering (KGQA) is the task of finding answers to questions posed in natural language, using triples present in a KG. Typically the following steps are followed in KGQA: 1) Objects of interest in the natural language question are detected and linked to the KG in a step called entity linking. 2) The relation between the objects is discovered and linked to the KG in a step called relation linking. 3) A

formal query, usually SPARQL[1], is formed with the linked entities and relations. The query is executed on the KG to fetch the answer.

Our focus in this work is the query building phase, henceforth referred to as KGQA semantic parsing. The motivation of our work stems from Banerjee, Nair, Kaur, et al. (2022), where minor vocabulary substitutions to handle non-printable special characters for T5 (Raffel et al., 2020) produced better results on the task of SPARQL semantic parsing. In this work, we extend the idea and replace the entire SPARQL vocabulary with alternate vocabularies.

As in Banerjee, Nair, Kaur, et al. (2022), we replace certain special characters in the SPARQL vocabulary, such as { , } with textual identifiers, as T5 is known to have problems dealing with these special characters (Banerjee, Nair, Kaur, et al., 2022). We call this a masked query, and in this work, we test the ability of the models to generate this masked query, given the natural language question as input.

A sample question, the original SPARQL query, and the corresponding masked query are as shown below (for the Wikidata KG (Vrandečić and Krötzsch, 2014)) :

Is it true that an Olympic-size swimming pool's operating temperature is equal to 22.4 ?

```
ASK WHERE
{
  wd:Q2084454 wdt:P5066 ?obj
  filter(?obj = 22.4)
}
```

```
ASK WHERE
OB
  ent0 rel0 ?obj
  filter ( ?obj = 22.4 )
CB
```

In the era of pre-trained Language Models (LMs) (Devlin et al., 2019; Raffel et al., 2020) it is common practice to fine-tune models on custom downstream datasets. This requires supervised training which results in modification of weights of the models using some training algorithm. More recently, the technique of prompting of language models (Brown et al., 2020; Shin et al., 2020) has been developed, which elicits the desired response from a LM through a task description and a few input-output examples. Brown et al. (2020) shows that such a strategy works better for larger models. It has however been observed that prompt design is brittle in behaviour and displays sensitivity to the exact phrase (Shin et al., 2020).

A more recent innovation is that of prompt tuning (Lester et al., 2021), where the task-specific prompt is learnt on a smaller external neural network. The gradients are computed and flow through the LM, but leave the weights of the LM itself unchanged. Instead, the weights of the prompt tuning network change and produce a custom and continuous prompt which produces the desirable response from the LM.

A similar method is prefix tuning (Li and Liang, 2021), which is known to perform better for generation tasks (F Ma et al., 2022). In this method, the original inputs and outputs are kept the same, but the input is pre-pended with a continuous prefix learnt

[1] https://www.w3.org/TR/rdf-sparql-query/

in the external network. This prefix allows the model to understand the exact task to be performed by it.

As primary contribution, in this work, we perform an analysis of how the complexity of output vocabularies affects the performance on the KGQA semantic parsing task for prefix and fine-tuned language models. Code and data can be found at https://github.com/debayan/sparql-vocab-substitution.

5.4 Related Work

A study of low-resource semantic parsing using prompt tuning was performed by Schucher et al. (2022) on the Top v2 (X Chen et al., 2020) and Overnight (Y Wang et al., 2015) datasets. Prompt tuning, while not the same as prefix tuning, still keeps the LM weights frozen while the prompts are learnt on an external network. In their experiments, they perform a single kind of vocabulary substitution but find no noticeable performance improvements. No specific study is made of the change in performance with vocabularies of varying complexities, which is a task we undertake. Another difference is that we perform experiments in the high-resource use case as opposed to low-resource.

Another work which is similar to ours is W Sun et al. (2022), where the authors experiment with prefix tuning on the task of semantic parsing, and find problems with non-standard vocabularies of logical forms. In their case, they work with the TOP v2 (X Chen et al., 2020) and PIZZA (Arkoudas et al., 2022) datasets. The keywords in those datasets consist of words joined by underscores (eg: IN:GET_REMINDER_DATA_TIME), which poses a problem for the sub-word tokenizer of the transformer based models. They find that fine tuning a model on these datasets outperforms prefix-tuning by a large margin. However, when they add the non-standard keywords to the tokenizer vocabulary and re-train the tokenizer to generate new embeddings for these keywords, fine tuning and prefix tuning perform at par. Our work is different in a few respects: firstly, due to the specific research focus of our group, we experiment with a semantic parsing dataset for KGQA, namely GrailQA (Gu, Kase, et al., 2021). Secondly, instead of retraining the tokenizer, we perform a simpler procedure of pre-processing the dataset by replacing the current vocabulary with a new vocabulary. We then train the models on this modified dataset, and as a post-processing step, substitute back the original vocabulary in place of the new vocabulary.

5.5 Prefix Tuning

Prefix tuning prepends a set of tunable weights to every key-value pair in the transformer attention. The transformer attention is represented as follows:

$$\text{attn}(Q, K, V) = \text{softmax}(\frac{Q \cdot K^\top}{\sqrt{d}})V \tag{5.1}$$

where the query Q, key K and value V are obtained through affine transformations on the input. d represents the model dimension. Prefix tuning modifies the transformer attention by adding tunable prefixes to K and V, thereby modifying K as $K' = [h_K; K]$ and V as $V' = [h_V; V]$. Here h_K and h_V represent the key prefix and the value prefix respectively.

	GrailQA					
	T5-Small		T5-Base			
	PT	FT	PT	FT	TSVS	ALFL
char8	74.03	86.57	82.65	86.72	306	263
char4	76.43	87.09	84.92	87.10	159	141
char2	83.29	91.49	89.83	92.30	90	87
char1	**84.89**	**92.13**	**91.24**	**92.61**	57	57
dictionary	82.57	91.95	90.93	92.48	49	44
original	67.10	74.08	73.06	74.45	124	125

Table 5.1: Exact match percentages for generated masked SPARQL queries. Best performance is always found in substituted vocabularies. For **char** settings, accuracy drops as vocabulary and query lengths increase. TSVS = Tokenizer specific vocabulary size, ALFL = Average logical form length, PT = Prefix Tuning, FT = Fine Tuning

Following Li and Liang (2021) we model these prefixes using a two layer MLP as follows:

$$\begin{aligned} h_K &= W_{K,2} f(W_{K,1} E + b_{K,1}) + b_{K,2} \\ h_V &= W_{V,2} f(W_{V,1} E + b_{V,1}) + b_{V,2} \end{aligned} \tag{5.2}$$

where $W \in \mathbb{R}^{d \times d}$ and $b \in \mathbb{R}^{d}$ are trainable weights and biases respectively. $E \in \mathbb{R}^{C \times d}$ is a trainable embedding matrix with C as the prefix length.

5.6 Models and Experimental Setup

We carry out prefix-tuning and fine-tuning experiments with two versions of the T5 model: namely T5-Small (60 million parameters) and T5-Base (220 million parameters). Questions are fed as input during training while masked SPARQL queries, as described in Section 5.3, are provided as labels for supervision.

For evaluation, we use the exact-match metric. A generated query is matched token by token, while ignoring white-spaces, to the gold query. The percentage of queries matched is reported.

5.6.1 Hyper-parameters and Implementation Details

Throughout our experiments, the prefix length is fixed to 50. For prefix tuning experiments we use the Adafactor (Shazeer and Stern, 2018) optimizer with a constant learning rate of 0.001. Fine-tuning experiments are optimized through AdamW (Loshchilov and Hutter, 2019) with a square root decay schedule, a maximum learning rate of 0.0015 and a linear warm-up of 5000 steps. Our code is implemented with HuggingFace Transformers[2] (Wolf et al., 2020) and OpenPrompt[3] (N Ding et al., 2022). T5-Small experiments were run on 12GB Nvidia GTX-1080 and RTX-2080 GPUs, and T5-Base experiments were run on 48GB Nvidia RTX-A6000. For fine-tuning, we run each training thrice with three separate seeds for 120 epochs each. For prompt tuning we do the same for 400 epochs. We report the inference results of these trained models on the test sets of the respective datasets.

[2] https://github.com/huggingface/transformers
[3] https://github.com/thunlp/OpenPrompt

5.7 Vocabulary

The original vocabulary of the GrailQA dataset consists of 48 words. The T5 tokenizer splits these words into 124 sub-words. This tokenizer specific vocabulary size (TSVS) is seen in the last column of Table 5.1. In the next column, the original average logical form (SPARQL query) length can be seen as 125 tokenized sub-words.

We wish to see how a new output vocabulary affects performance, and as a result, we construct a set of special vocabularies and substitute them in-place of the original SPARQL vocabulary. With reference to the settings in Table 5.1, each vocabulary is as described below:

original The masked SPARQL queries remain as they are. No replacement of the original SPARQL keywords is made with an alternate vocabulary.

dictionary The SPARQL keywords are replaced with a vocabulary of English words. For example, SELECT may be replaced with DOG, [may be replaced with CAT etc. During the pre-training phase a LM is likely to have seen such words far more frequently than the SPARQL keywords. This mode tests how the model behaves when the output vocabulary is comprised of well known English words.

char1 The SPARQL keywords are replaced with a single character of the English alphabet, for example, SELECT is replaced with A, WHERE is replaced with B. Additionally, numerical digits from 1-9 are used, and if the size of vocabulary demands more, we add single length special characters, such as * and $.

char2, **char4** and **char8** settings apply vocabulary substitution of 2, 4 and 8 character lengths chosen randomly, constituted from the characters A-Z and digits 0-9. For example, a typical **char8** substitution would be SELECT replaced by ATYZGFSD. This setting is designed to test the behaviour of the models when asked to produce more number of tokens per original-vocabulary word. A sample of a question, the SPARQL and the corresponding substitutions is provided in the Appendix in Table 5.2.

5.8 Datasets

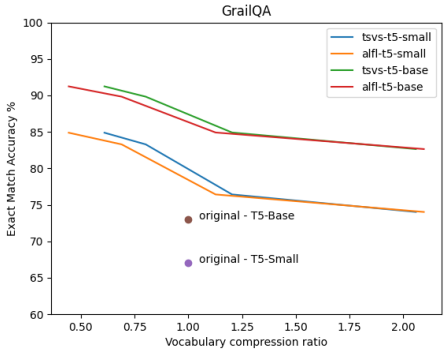
For our experiments, we require a dataset which contains a mapping of natural language questions to their corresponding logical forms and is large in size, since we test the high resource use-case.

GrailQA [4] is based on the Freebase knowledge graph (Bollacker et al., 2008) and consists of 64,331 questions designed to test three levels of generalisation, ie, i.i.d, compositional and zero-shot. For our purposes, we split the train set itself to three parts, since we are not interested in testing compositional generalisation aspects of the test set of this dataset. We are left with the following configuration: test: 8868, dev: 4434, train: 31035.

5.9 Analysis

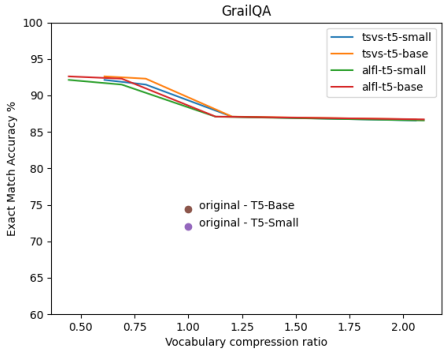
As seen in Table 5.1, the best performance for prefix and fine tuning is achieved for substituted vocabularies. The original vocabulary lags behind in general, which points to the finding, that the choice of an appropriate vocabulary improves performance

[4] https://dki-lab.github.io/GrailQA/



(a)

Figure 5.1: Prefix tuning accuracy drops as vocabulary and query lengths increase for **char** settings. TSVS = Tokenizer specific vocabulary size, ALFL = Average logical form length



(a)

Figure 5.2: Fine-tuning accuracy drop is more gradual when compared to prefix tuning, and the performance of T5-Small and T5-Base are similar. TSVS = Tokenizer specific vocabulary size, ALFL = Average logical form length

for semantic parsing. Further, among the substituted vocabularies, the setting **char8** performs the worst, which signifies the adverse role of the extra decoding load of this vocabulary on the performance of the model.

This finding is different from that of Schucher et al. (2022), who find their `in-vocab` setting performing no better overall. They attribute it to the substitutions possibly masking the meanings of the intents, for their given dataset. On the contrary, we find significant gains for GrailQA. It must be noted however, that we perform high-resource prefix tuning while they perform low-resource prompt tuning, and hence results may differ.

As seen in Figure 5.1, for the **char** settings, as the size of vocabulary increases, the prefix tuning accuracy drops. In the said figure, we define vocabulary compression ratio as the size of the new vocabulary divided by the size of the original vocabulary. Apart from vocabulary size, the query length also matters. We dual-define vocabulary compression ratio as the size of query length after substitution of new vocabulary divided by size of original query length, and plot on the same graph.

When compared to the fine-tuning plot (Figure 5.2), prefix tuning has a steeper drop in accuracy, and the performance for T5-Small and T5-Base vary more significantly. It leads to the finding that fine-tuning is less sensitive to vocabulary changes, and the difference in model sizes between T5-Small and T5-Base also seems to matter less.

In Figures 5.1 and 5.2, it can be seen that the **original** setting for the masked SPARQL vocabularies produce accuracies which are below the **char** family vocabulary curves. It suggests that vocabulary compression ratio alone is not a deciding factor in accuracy. If the vocabulary family changes from SPARQL to characters, there is an initial shift in accuracy, and after that the complexity of the character vocabulary further affects the accuracy.

In Table 5.1, the **dictionary** setting performs slightly worse than the **char1** setting, although it has lower TSVS and ALFL. This suggests that the vocabulary size and query length are not the only factors that affect the eventual accuracy. Perhaps the frequency of the tokens seen by the model during the pre-training task plays a role. It is likely that the model has encountered, during pre-training, single characters a far larger number of times than the words used in **dictionary** vocabulary.

5.10 Error Analysis

We performed an error analysis on a sample of 100 randomly selected questions which produced an incorrect output. In the **original** setting, roughly 50% errors were due to the presence of non-printable characters in the query (eg: ^). We found that in the initial masked query, while we had replaced some non-printable characters in the pre-processing stage (eg: {, }), we had not managed to replace the full set of non-printable characters. The original T5 paper mentions curly braces as one of the class of tokens that are not present in the pre-training corpus, however, a comprehensive list of the tokens that do not work with T5, or work with limited efficiency, is not available. In this scenario, it seems that a better approach is to replace the entire vocabulary with one that is entirely known to T5, for example, English words. When comparing errors made by **original**, that were fixed by **dictionary** and **char1**, we observed that roughly 30% of the cases were of variable placement, where the variable placeholders like `ent0`, `rel0` were found to be in the wrong order in the output query in the **original** setting. Rest of the corrections belonged to the category of syntax errors. This points to the finding that alternate vocabularies improve the ability of T5 to correctly produce logical forms from a semantic perspective.

To analyse the effect of increasing complexity of vocabulary, we compare 100 randomly selected errors made by **char8** with **char2**. In both these settings, no character is non-printable, and the only errors are either syntax errors, variable placement errors, structural errors or intent errors. Out of the 100 questions, 90 were found to be correct in **char2** setting. In the remaining 90 in the **char8** setting, the highest proportion of errors belonged to syntax (where the query is malformed). The next most prominent class of errors belonged to variable placement, followed by structural errors (eg: two triples instead of three). The major takeaway from this analysis is that for **char2** there were no syntax errors, while in **char8** there are a significant number of such errors.

5.11 Conclusion

In this work we carried out experiments with new output vocabularies, where we carefully substituted the original members of the vocabulary with the new ones. We found that when the original SPARQL vocabulary is replaced with words from an

alternate vocabulary closer to the T5 tokenizer vocabulary, the model consistently perform better.

As a contribution, we believe that our findings will enable researchers in the field of semantic parsing to deploy smaller models with a modified vocabulary and still find satisfactory performance. This would, in the longer term, lead to energy savings.

As future work, we would like to explore the behaviour of the same models in more depth using attention maps. Moreover, the significant shift in initial performance on changing vocabulary from **original** to **char** and **dictionary** demands further investigation. Similarly, the relatively lower performance of the **dictionary** setting when compared to **char1** setting, in spite of having lower tokenized vocabulary size (TSVS) needs to be investigated further. Perhaps sub-words which are seen more frequently during pre-training task of the LM perform better when substituted into the semantic parsing output vocabulary.

5.12 Limitations

We found that prefix tuning takes much longer to converge when compared to fine tuning, and for T5-Base, it takes around 10 days on a 48 GB GPU to complete tuning for a single setting in Table 5.1. Due to limitation of resources and with an aim to save energy, we did not conduct experiments with larger models such as T5-Large, T5-XL etc. We also did not perform experiments with smaller splits of the same datasets, which could have given further insights on how model performance varies when training data size is less.

5.13 Samples

	GrailQA
Question	Military airfield is the type for what airport ?

SPARQL	`SELECT DISTINCT ?x0 WHERE {`
	`?x0 :type.object.type :aviation.airport .`
	`VALUES ?x1 { :m.0199qf }`
	`?x0 :aviation.airport.airport_type ?x1 .`
	`FILTER ( ?x0 != ?x1 )`
	`}`

Masked Query (**original** setting)	`SELECT DISTINCT ?x0 WHERE OB`
	`?x0 :type.object.type rel0 .`
	`VALUES ?x1 OB ent0 CB`
	`?x0 rel1 ?x1 .`
	`FILTER ( ?x0 != ?x1 )`
	`CB`

dictionary	banana compound boy nation rain
	  boy catastrophe elementary flower
	  teeth today rain jacket case
	  boy fog today flower
	  duck folk boy chart today concede
	case

char1	- 1 A Y $
	  A : O %
	  L J $ G S
	  A | J %
	  0 M A + J X
	S

char2	UY SJ 0X 6L VZ
	  0X 5G JO SE
	  5Z QB VZ QJ 8O
	  0X FT QB SE
	  RU 2K 0X WY QB I5
	8O

char4	53IY 3UQZ JKMQ CEK2 5DZV
	  JKMQ KRDN 1G8E ZC5C
	  5ILL 3JBD 5DZV X5XB YMG5
	  JKMQ ZVGC 3JBD ZC5C
	  87O2 DE3Z JKMQ TU76 3JBD 049K
	YMG5

char8	WDEUTG57 L741BHJP ORWDXYPH 6L05N8AS ZLZXSARH
	  ORWDXYPH K4GR9TPQ 797G3PGO V13Y1EFE
	  PQMAIPQ4 MLN1V72G ZLZXSARH KPHC8I2N WG0XRTYG
	  ORWDXYPH ZF82YUH8 MLN1V72G V13Y1EFE
	  41O2LA2M F1SANW03 ORWDXYPH 4R26K1BW MLN1V72G TD9BSKSN
	WG0XRTYG

Table 5.2: An example of a question from GrailQA, with the corresponding SPARQL query, and how they look once new vocabularies are substituted.

6

DBLP-QuAD: A Question Answering Dataset over the DBLP Scholarly Knowledge Graph

6.1 Bibliographic Information

Debayan Banerjee, Sushil Awale, Ricardo Usbeck and Chris Biemann. 2023. DBLP-QuAD: A Question Answering Dataset over the DBLP Scholarly Knowledge Graph. In Proceedings of the 13th International Workshop on Bibliometric-enhanced Information Retrieval. Dublin, Ireland. Pages 37-51.
https://doi.org/10.48550/arXiv.2303.13351

6.2 Abstract

In this work we create a question answering dataset over the DBLP scholarly knowledge graph (KG). DBLP is an on-line reference for bibliographic information on major computer science publications that indexes over 4.4 million publications published by more than 2.2 million authors. Our dataset consists of 10,000 question answer pairs with the corresponding SPARQL queries which can be executed over the DBLP KG to fetch the correct answer. DBLP-QuAD is the largest scholarly question answering dataset.

6.3 Introduction

Over the past decade, knowledge graphs (KG) such as Freebase (Bollacker et al., 2008), DBpedia (Lehmann, Isele, et al., 2015), and Wikidata (Vrandečić and Krötzsch, 2014) have emerged as important repositories of general information. They store facts about the world in the linked data architecture, commonly in the format of <subject predicate object> triples. These triples can also be visualised as node-edge-node molecules of a graph structure. Much interest has been generated in finding ways to retrieve

information from these KGs. Question Answering over Knowledge Graphs (KGQA) is one of the techniques used to achieve this goal. In KGQA, the focus is generally on translating a natural language question to a formal logical form. This task has, in the past, been achieved by rule-based systems (Dubey, Dasgupta, et al., 2016). More recently, neural network and machine learning based methods have gained popularity (Chakraborty et al., 2019).

A scholarly KG is a specific class of KGs that contains bibliographic information. Some well known scholarly KGs are the Microsoft Academic Graph[1], OpenAlex[2], ORKG[3] and DBLP[4]. DBLP caters specifically to the bibliography of computer science, and as a result, it is smaller in size than other scholarly KGs. We decided to build our KGQA dataset over DBLP due to its focused domain and manageable size so that we could concentrate on adding complexity to the composition of the KGQA dataset itself.

Datasets are important, especially for ML-based systems, because such systems often have to be trained on a sample of data before they can be used on a similar test set. To this end, several KGQA datasets exist (Perevalov et al., 2022). However, not all datasets contain a mapping of natural language questions to the logical form (e.g. SPARQL, λ-calculus, S-expression). Some simply contain the question and the eventual answer. Such datasets can not be used to train models in the task of semantic parsing.

In this work, we present a KGQA dataset called DBLP-QuAD, which consists of 10,000 questions with corresponding SPARQL queries. The question formation process begins with human-written templates, and later, we machine-generate more questions from these templates. DBLP-QuAD consists of a variety of simple and complex questions and also tests the compositional generalisation of the models. DBLP-QuAD is the largest scholarly KGQA dataset being made available to the public[5].

6.4 Related Work

ORKG-QA benchmark (Jaradeh et al., 2020) is the first scholarly KGQA dataset grounded to ORKG. The dataset was prepared using the ORKG API and focuses on the content of academic publications structured in comparison tables. The dataset is relatively small in size with only 100 question-answer pairs covering only 100 research publications.

Several other QA datasets exist, both for IR-based QA (Rajpurkar et al., 2018; Kwiatkowski et al., 2019) and KGQA (Trivedi et al., 2017; Sen et al., 2022) approaches. Several different approaches have been deployed to generate the KGQA datasets. These approaches range from manual to machine generation. However, most datasets lie in between and use a combination of manual and automated process.

A clear separation can be created between datasets that contain logical forms and those that do not. Datasets that do not require logical forms can be crowd-sourced and such datasets are generally large in size. Crowd sourcing is generally not possible for annotating logical forms because this task requires high domain expertise and it is not easy to find such experts on crowd sourcing platforms. We focus on datasets that contain logical forms.

[1] https://www.microsoft.com/en-us/research/project/microsoft-academic-graph/
[2] http://openalex.org/
[3] https://orkg.org/
[4] https://dblp.org/
[5] https://doi.org/10.5281/zenodo.7643971

Free917 and QALD (Cai and Yates, 2013; Usbeck et al., 2017) datasets were created manually by domain experts, however, their sizes are relatively small (917 and 806 respectively).

WebQuestionsSP and ComplexWebQuestions (Yih, M Richardson, et al., 2016; Talmor and Berant, 2018) are developed using exisiting datasets. WebQuestionsSP is a semantic parsing dataset developed by using questions from WebQuestions (Berant, Chou, et al., 2013b). Yih, M Richardson, et al. (2016) developed a dialogue-like user interface which allowed five expert human annotators to annotate the data in stages.

ComplexWebQuestions is a collection of 34,689 complex question paired with answers and SPARQL queries grounded to Freebase KG. The dataset builds on WebQuestionsSP by sampling question-query pairs from the dataset and automatically generating questions and complex SPARQL queries with composition, conjunctions, superlatives, and comparatives functions. The machine generated questions are manually annotated to natural questions and validated by 200 AMT crowd workers.

The OVERNIGHT (ON) approach is a semantic parsing dataset generation framework introduced by Y Wang et al. (2015). In this approach, the question-logical form pairs are collected with a three step process. In the first step, the logical forms are generated from a KG. Secondly, the logical forms are converted automatically into canonical questions. These canonical questions are grammatically incorrect but successfully carry the semantic meaning. Lastly, the canonical questions are converted into natural forms via crowdsourcing. Following are some of the datasets developed using this approach.

GraphQuestions (Su et al., 2016) consists of 5,166 natural questions accompanied by two paraphrases of the original question, an answer, and a valid SPARQL query grounded against the Freebase KG. GraphQuestions uses a semi-automated three-step algorithm to generate the natural questions for the KG.

LC-QuAD 1.0 (Trivedi et al., 2017) is another semantic parsing dataset for the DBpedia KG. LC-QuAD 1.0 is relatively larger in size with 5,000 natural language English questions and corresponding SPARQL queries. The generation process starts with the set of manually created SPARQL query templates, a list of seed entities, and a whitelist of predicates. Using the list of seed entities, two-hop subgraphs from DBpedia are extracted. The SPARQL query templates consist of placeholders for both entities and predicates which are instantiated using triples from the subgraph. These SPARQL queries are then used to instantiate natural question templates which form the base for manual paraphrasing by humans.

LC-QuAD 2.0 (Dubey, Banerjee, Abdelkawi, et al., 2019) is the second iteration of LC-QuAD 1.0 with 30,000 questions, their paraphrases and their corresponding SPARQL queries compatible with both Wikidata and DBpedia KGs. Similar to LC-QuAD 1.0, in LC-QuAD 2.0 a sub-graph is generated using seed entities and a SPARQL query template is selected based on whitelist predicates. Then, the query template is instantiated using the sub-graph. Next, a template question is generated from the SPARQL query which is then verbalised and paraphrased by AMT crowd workers. LC-QuAD 2.0 has more questions and more variation compared to LC-QuAD 1.0 with paraphrases to the natural questions.

GrailQA (Gu, Kase, et al., 2021) extends the approach in Su et al. (2016) to generate 64,331 question-S-expression pairs grounded to the Freebase Commons KG. Here, S-expression are linearized forms of graph queries. Query templates extracted from graph queries generated from the KG are used to generate canonical logical forms grounded to compatible entities. The canonical logic forms are then validated by a graduate student if they represent plausible user query or not. Next, another graduate student annotated

the validated canonical logic form with a canonical question. Finally, 6,685 Amazon Mechanical Turk workers write five natural paraphrases for each canonical question which are further validated by multiple independent crowd workers.

KQA Pro (Cao et al., 2022) is a large collection of 117,000 complex questions paired with SPARQL queries for the Wikidata KG. KQA Pro dataset also follows the OVERNIGHT approach where firstly facts from the KG are extracted. Next, canonical questions are generated with corresponding SPARQL queries, ten answer choices and a golden answer. The canonical questions are then converted into natural language with paraphrases using crowd sourcing.

CFQ (Keysers et al., 2020) (Compositional Freebase Questions) is a semantic parsing dataset developed completely using synthetic generation approaches that consists of simple natural language questions with corresponding SPARQL query against the Freebase KG. CFQ contains 239,357 English questions which are generated using hand-crafted grammar and inference rules with a corresponding logical form. Next, resolution rules are used to map the logical forms to SPARQL queries. The CFQ dataset was specifically designed to measure compositional generalization.

In this work, we loosely follow the OVERNIGHT approach to create a large scholarly KGQA dataset for the DBLP KG.

6.5 DBLP KG

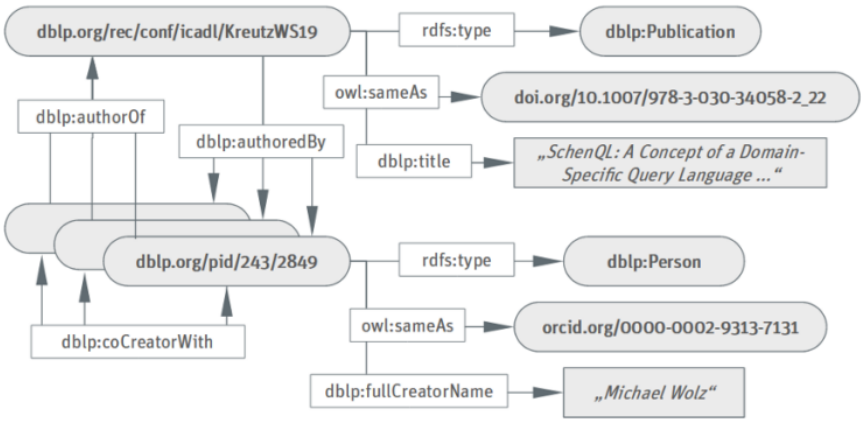


Figure 6.1: Example of entries in the DBLP KG with its schema

DBLP, which used to stand for Data Bases and Logic Programming[6], was created in 1993 by Michael Ley at the University of Trier, Germany (Ley, 2002). The service was originally designed as a bibliographic database for research papers and proceedings from the fields of database systems and logic programming. Over time, the service has grown in size and scope, and today includes bibliographic information on a wide range of topics within the field of computer science. The DBLP RDF data models a person-publication graph shown in Figure 6.1.

The DBLP KG contains two main entities: *Person* and *Publication*, where as other metadata such as journal and conferences, affiliation of authors are currently only string literals. Henceforth, we use the term *person* and *creator* interchangeably. At

[6] https://en.wikipedia.org/wiki/DBLP

the time of its release, the RDF dump consisted of 2,941,316 person entities, 6,010,605 publication entities, and 252,573,199 RDF triples. DBLP currently does not provide a SPARQL endpoint but the RDF dump can be downloaded and a local SPARQL endpoint such as Virtuoso Server can be setup to run a SPARQL query against the DBLP KG.

The live RDF data model on the DBLP website follows the schema shown in Figure 6.1. However, the RDF snapshots available for download have the *coCreatorWith* and *authorOf* predicates missing. Although these predicates are missing, the *authoredBy* predicate can be used to derive the missing relations. DBLP-QuAD is based on the DBLP KG schema of the downloadable RDF graph.

6.6 Dataset Generation Framework

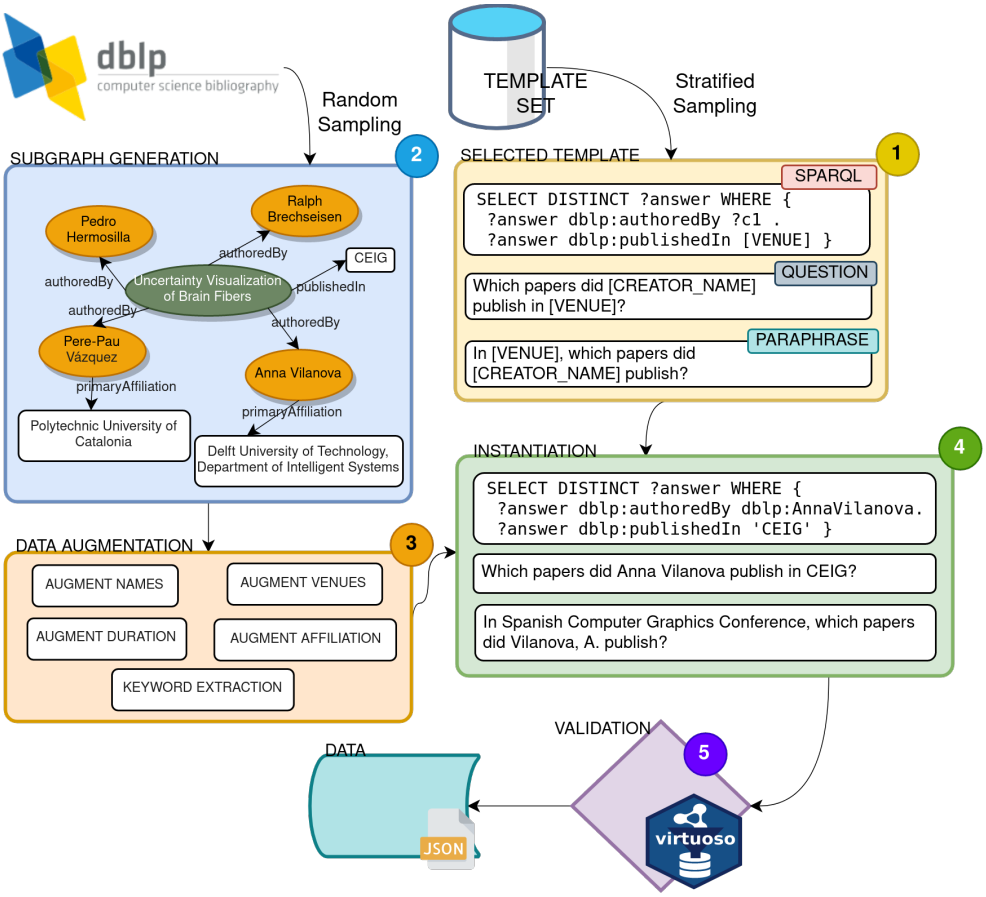


Figure 6.2: Motivating Example. The generation process starts with (1) selection of a template tuple followed by (2) subgraph generation. Then, literals in subgraph are (3) augmented before being used to (4) instantiate the selected template tuple. The generated data is (5) filtered based on if they produce answers or not.

In this work, the aim is to generate a large variety of scholarly questions and corresponding SPARQL query pairs for the DBLP KG. Initially, a small set of templates T containing a SPARQL query template s_t and a few semantically equivalent natural language question templates Q_t are created. The questions and query templates are created such that they cover a wide range of scholarly metadata user information need while also being answerable using a SPARQL query against the DBLP KG. Next, we

synthetically generate a large set of question-query pairs (q_i, s_i) suitable for training a neural network semantic parser.

The core methodology of the dataset generation framework encompasses instantiating the templates using literals of subgraphs sampled from the KG. Moreover, to capture different representations of the literal values from a human perspective, we randomly mix in different augmentations of these textual representations. The dataset generation workflow is shown in Figure 6.2.

6.6.1 Templates

The first step in the dataset generation process starts with the creation of a template set. After carefully analyzing the ontology of the DBLP KG, we manually wrote 98 pairs of valid SPARQL query templates and a set of semantically equivalent natural language question templates. The template set was written by one author and verified for correctness by another author. The query and question templates consist of placeholder markers instead of URIs, entity surface forms or literals. For example, in Figure 6.2 (Section 1), the SPARQL query template includes the placeholders ?$c1$ and [$VENUE$] for DBLP person URI and venue literal respectively. Similarly, the question templates include placeholders [$CREATOR\_NAME$] and [$VENUE$] for creator name and venue literal respectively. The template set covers the two entities creator and publication, and additionally the foreign entity bibtex type. Additionally, they also cover the 11 different predicates of DBLP KG.

The template set consists of template tuples. A template tuple $t = (s_t, Q_t, E_t, P_t)$ is composed of a SPARQL query template s_t, a set of semantically equivalent natural language question templates Q_t, a set of entity placeholders E_t and a set of predicates P_t used in s_t. We also add a boolean indicating whether the query template is temporal or not and another boolean indicating whether to use or not use the template while generating *train* dataset. Each template tuple contains between four and seven paraphrased question templates offering wide linguistic diversity. While most of the question templates use the *"Wh-"* question keyword, we also include instruction-style paraphrases.

We group the template tuples as creator-focused or publication-focused ϵ and further group them by query types δ. We have 10 different query types and they include Single Fact, Multiple Facts, Boolean, Negation, Double Negation, Double Intent, Union, Count, Superlative/Comparative, and Disambiguation. The question types are discussed in Section 6.6.6 with examples. The distribution of templates per entity and query type is shown in Table 6.1. During dataset generation, for each data instance we sample a template tuple from the template set using stratified sampling maintaining equal distribution of entity types and query types.

6.6.2 Subgraph generation

The second part of the dataset generation framework is subgraph generation. Given a graph $G = (V, E)$ where V are the vertices, and E are edges, we draw a subgraph $g = (v, e)$ where $v \subset V$, $e \subset E$. For the DBLP KG, V are the creator and publication entity URIs or literals, and the E are the predicates of the entities.

The subgraph generation process starts with random sampling of a publication entity v_i from the DBLP KG. We only draw from the set of publication entities as the RDF snapshot available for download has $authorOf$ and $coCreatorWith$ predicates missing

Query Type	Creator-focused	Publication-focused	Total
Single Fact	5	5	10
Multiple Facts	7	7	14
Boolean	6	6	12
Negation	4	4	8
Double Negation	4	4	8
Double Intent	5	4	9
Union	4	4	8
Count	6	5	11
Superlative/Comparative	6	6	12
Disambiguation	3	3	6
Total	50	48	98

Table 6.1: Total number of template tuples per query type grouped by entity type

for creator entity. As such, a subgraph centered on a creator entity would not have end vertices that can be expanded further. With the sampled publication entity v_i, we iterate through all the predicates e to extract creator entities v' as well as the literal values. We further, expand the creator entities and extract their literal values to form a two-hop subgraph $g = (v, e)$ as shown in Figure 6.2 (Section 2).

6.6.3 Template Instantiation

Using the generated subgraph and the sampled template tuple, the template tuple is instantiated with entity URIs and literal values from the subgraph. In the instantiation process, a placeholder marker in a string is replaced by the corresponding text representation.

For the SPARQL query template s_t, we instantiate the creator/publication placeholder markers with DBLP creator/publication entity URIs or literal values for affiliation and conference or journals to create a valid SPARQL query s that returns answers when run against the DBLP KG SPARQL endpoint.

In case of natural language question templates, we randomly sample two from the set of question templates $q_t^1, q_t^2 \in Q_T$, and instantiate each using only the literal values from the subgraph to form one main natural language question q^1 and one natural language question paraphrase q^2. In natural language, humans can write the literal strings in various forms. Hence to introduce this linguistic variation, we randomly mix in alternate string representations of these literal values in both natural language questions. The data augmentation process allows us to add heuristically manipulated alternate literal representations to the natural questions. A example of an instantiated template is shown in Figure 6.2 (Section 3).

6.6.4 Data Augmentation

For the template instantiation process, we perform simple string manipulations to generate alternate literal representations. Then, we randomly select between the original literal representation and the alternate representation to instantiate the natural language questions. For each literal type, we apply different string manipulation techniques which we describe below.

Names: For names we generate four different alternatives involving switching parts of names or keeping only initials of the names. Consider the name *John William Smith* for which we produce *Smith, John William*, *J. William Smith*, *John W. Smith*, and *Smith, J. William*.

Venues: Venues can be represented using either its short form or its full form. For example, *ECIR* or *European Conference on Information Retrieval*. In DBLP venues are stored in its short form. We use a selected list of conference and journals[7] containing the short form and its equivalent full form to get the full venue names.

Duration: About 20% of the templates contain temporal queries, and some of them require dummy numbers to represent duration. For example, the question *"In the last five years, which papers did Mante S. Nieuwland publish?"* uses the dummy value *five*. We randomly select between the numerical representation and the textual representation for the dummy duration value.

Affiliation: In natural language questions, only the institution name is widely used to refer to the affiliation of an author. However, the DBLP KG uses the full address of an institution including city and country name. Hence, using RegeEx we extract the institution names and randomly select between the institution name and the full institution address in the instantiation process.

Keywords: For disambiguation queries, we do not use the full title of a publication but rather a part of it by extracting keywords. For this purpose, we use SpaCy's Matcher API[8] to extract noun phrases from the title.

6.6.5 Dataset Generation

For each data instance d_i, we sample 2 subgraphs (*SampleSubgraph(G,2)*) and instantiate a template tuple t_i (*Instantiate(t_i, g_1, g_2, x)*). We sample 2 subgraphs as some template tuples require to be instantiated with two publication titles. Each data instance $d_i = (s_i, q_i^1, q_i^2, E_i, P_i, y, z)$ comprises of a valid SPARQL query s_i, one main natural language question q_i^1, one semantically equivalent paraphrase of the main question q_i^2, a list of entities E_i used in s_i, a list of predicates P_i used in s_i, a Boolean indicating whether the SPARQL query is temporal or not y, and another Boolean informing whether the SPARQL query is found only in *valid* and *test* sets z. We generate an equal number n of questions for each entity group ϵ equally divided for each query type δ.

To foster a focus on generalization ability, we manually marked 20 template tuples to withhold during generation of the *train* set. However, we use all the template tuples in the generation of *valid* and *test* sets. Furthermore, we also withhold 2 question templates when generating *train* questions but use all question templates when generating *valid* and *test* sets. This controlled generation process allows us to withhold some entity classes, predicates and paraphrases from *train* set. Our aim with this control is to create a scholarly KGQA dataset that facilitates development of KGQA models that adhere to *i.i.d*, *compositional*, and *zero-shot* (Gu, Kase, et al., 2021) generalization.

Further, we validate each data instance d_i by running the SPARQL query s_i against the DBLP KG via a Virtuoso SPARQL endpoint[9]. We filter out data instances for which the SPARQL query is invalid or generates a blank response. A SPARQL query may

[7] http://portal.core.edu.au/conf-ranks/?search=&by=all&source=CORE2021&sort=atitle&page=1

[8] https://spacy.io/api/matcher/

[9] https://docs.openlinksw.com/virtuoso/whatisvirtuoso/

Algorithm 1: Dataset Generation Process

```
GenerateDataset (T, x, N, G)
    inputs : template set T; dataset set to generate x; size of dataset to generate
             N; KG to sample subgraphs from G;
    output : dataset D;
    D ← ∅;
    n ← (N/|ϵ|)/|δ|;
    foreach e ∈ ϵ do
        foreach s ∈ δ do
            i ← 0;
            T_es ← T[e][s];
            if x == train then
                T_es ← Filter(T_es, test_only == True)
            while i < n do
                g_1, g_2 ← SampleSubgraph(G, 2);
                t_i ← random.sample(T_es);
                d_i ← Instantiate(t_i, g_1, g_2, x);
                answer ← Query(d_i);
                if answer then
                    D ← d_i;
                    i ← i + 1;
    return D
```

generate a blank response if the generated subgraphs have missing literal values. In the DBLP KG, some of the entities have missing literals for predicates such as *primaryAffiliation*, *orcid*, *wikidata*, and so on. Additionally, we also store the answers produced by the SPARQL query against the DBLP KG formatted according to https://www.w3.org/TR/sparql11-results-json/. The dataset generation process is summarized in Algorithm 1.

6.6.6 Types of Questions

The dataset is composed of the following question types. The examples shown here are hand-picked from the dataset.

- **Single fact**: These questions can be answered using a single fact. For example, "What year was 'SIRA: SNR-Aware Intra-Frame Rate Adaptation' published?"
- **Multiple facts**: These questions require connecting two or more facts to answer. For example, "In SIGCSE, which paper written by Darina Dicheva with Dichev, Christo was published?"
- **Boolean**: These questions answer where a given fact is true or false. We can also add negation keywords to negate the questions. For example, "Does Szeider, Stefan have an ORCID?"

- **Negation**: These questions require to negate the answer to the Boolean questions. For example, "Did M. Hachani not publish in ICCP?"
- **Double negation**: These questions require to negate the Boolean question answers twice which results. For example, "Wasn't the paper 'Multi-Task Feature Selection on Multiple Networks via Maximum Flows' not published in 2014?"
- **Count**: These questions pertain to the count of occurrence of facts. For example, "Count the authors of 'Optimal Symmetry Breaking for Graph Problems' who have Carnegie Mellon University as their primary affiliation."
- **Superlative/Comparative**: Superlative questions ask about the maximum and minimum for a subject and comparative questions compare values between two subjects. We group both types under one group. For example, "Who has published the most papers among the authors of 'k-Pareto optimality for many-objective genetic optimization'?"
- **Union** questions cover a single intent but for multiple subjects at the same time. For example, "List all the papers that Pitas, Konstantinos published in ICML and ISCAS."
- **Double intent** questions poses two user intentions, usually about the same subject. For example, "In which venue was the paper 'Interactive Knowledge Distillation for image classification' published and when?"
- **Disambiguation** questions requires identifying the correct subject in the question. For example, "Which author with the name Li published the paper about Buck power converters?"

6.7 Dataset Statistics

DBLP-QuAD consists of 10,000 unique question-query pairs grouped into *train*, *valid* and *test* sets with a ratio of *7:1:2*. The dataset covers 13,348 creators and publications, and 11 predicates of the DBLP KG. For each query type in Table 6.1, the dataset includes 1,000 question-query pairs each of which is equally divided as creator-focused or publication-focused. Additionally, among the questions in DBLP-QuAD, 2,350 are temporal questions.

Linguistic Diversity. In DBLP-QuAD, a natural language question has an average word length of 17.32 words and an average character length of 114.1 characters. Similarly, a SPARQL query has an average vocab length of 12.65 and an average character length of 249.48 characters. Between the natural language question paraphrases, the average Jaccard similarity for unigram and bigram are 0.62 and 0.47 (with standard deviations of 0.22 and 0.24) respectively. The average Levenshtein edit distance between them is 32.99 (with standard deviation of 23.12). We believe the metrics signify a decent level of linguistic diversity.

Entity Linking. DBLP-QuAD also presents challenging entity linking with data augmentation performed on literals during the generation process. The augmented literals present more realistic and natural representation of the entity surface forms and literals compared to the entries in the KG.

Generalization. In the *valid* set 18.9% and in the *test* set 19.3% of instances were generated using the withheld templates. Hence, these SPARQL query templates and natural language question templates are unique to the *valid* and *test* sets. Table 6.2 shows the percent of questions with different levels of generalization in the *valid* and *test* sets of the dataset.

Dataset	I.I.D	Compositional	Zero-shot
Valid	82.8%	13.6%	3.6%
Test	81.2%	15.1%	3.8%

Table 6.2: Percent of questions with different levels of generalization in the *valid* and *test* sets of DBLP-QuAD

6.8 Semantic Parsing Baseline

To lay the foundation for future work on DBLP-QuAD, we also release baselines using the recent work by Banerjee, Nair, Kaur, et al. (2022), where a pre-trained T5 model is fine-tuned (Raffel et al., 2020) on the LC-QuAD 2.0 dataset.

Following Banerjee, Nair, Kaur, et al. (2022), we assume the entities and the relations are linked, and only focus on query building. We formulate the source as shown in Figure 6.3, where for each natural language question a prefix "**parse text to SPARQL query**:" is added. The source string is further concatenated with entity URIs and relation schema URIs separated by a special token [*SEP*]. The target text is the corresponding SPARQL query which is padded with the tokens $< s >< /s >$. We also make use of the sentinel tokens provided by T5 to represent the DBLP prefixes e.g. *<extra_id_1>* denotes the prefix *https://dblp.org/pid/*, SPARQL vocabulary and symbols. This step helps the *T5-tokenizer* to correctly fragment the target text during inference.

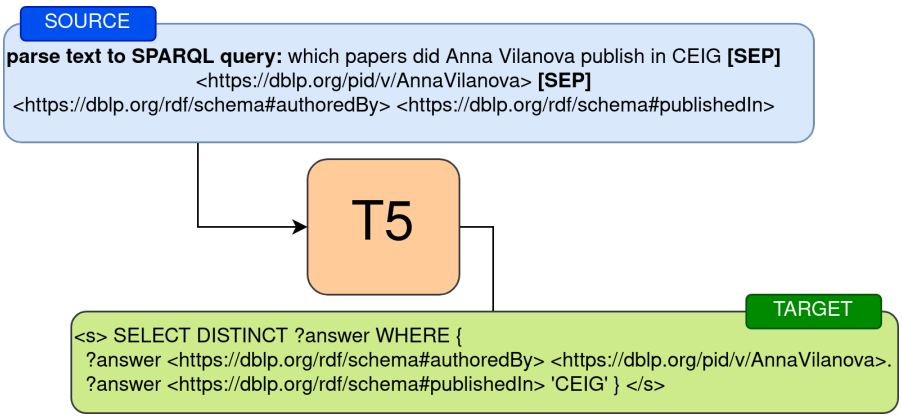


Figure 6.3: Representation of source and target text used to fine-tune the T5 model

We fine-tune *T5-Base* and *T5-Small* on DBLP-QuAD train set with a learning rate of *1e-4* for 5 epochs with an input as well as output text length of 512 and batch size of 4.

6.8.1 Experiment Results

We report the performance of the baseline model on the DBLP-QuAD test set. Firstly, we report on the exact-match between the gold and the generated SPARQL query. For

the exact-match accuracy we compare the generated and the gold query token by token after removing whitespaces. Next, for each SPARQL query on the test set, we run both the gold and and the query generated by the T5 baseline models using Virtuoso SPARQL endpoint to fetch answers from the DBLP KG. Based on the answers collected, we report on the F1 score. The results are reported on Table 6.3.

Evaluation metrics	T5-Small	T5-Base
Exact-match Accuracy	0.638	0.813
F1 Score	0.721	0.868

Table 6.3: Evaluation results of fine-tuned T5 to DBLP-QuAD

6.9 Limitations

One of the drawbacks of our dataset generation framework is that natural questions are synthetically generated. (CFQ (Keysers et al., 2020) has a similar limitation.) Although the question templates were human-written, only two people (authors of the paper) worked on the creation of the question templates and was not crowd sourced from a group of researchers. Additionally, the questions are generated by drawing data from a KG. Hence, the questions may not perfectly reflect the distribution of user information need. However, the machine-generation process allows for programmatic configuration of the questions, setting question characteristics, and controlling dataset size. We utilize the advantage by programmatically augmenting text representations and generating a large scholarly KGQA with complex SPARQL queries.

Second, in generating *valid* and *test* sets, we utilize additional 19 template tuples which account for about 20% of the template set. Therefore, the syntactic structure for 80% of the generated data in *valid* and *test* would already be seen in the train set resulting in test leakage. However, to limit the leakage on 80% of the data, we withhold 2 question templates in generating the *train* set. Moreover, the data augmentation steps carried out would also add challenges in the *valid* and *test* sets.

Another shortcoming of DBLP-QuAD is that the paper titles do not perfectly reflect user behavior. When a user asks a question, they do not type in the full paper title and also some papers are popularly known by a different short name. For example, the papers "Language Models are Few-shot Learners" and "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" are also known as "GPT-3" and "BERT" respectively. This is a challenging entity linking problem which requires further investigation. Despite the shortcomings, we feel the large scholarly KGQA dataset would ignite more research interest in scholarly KGQA.

6.10 Conclusion

In this work, we presented a new KGQA dataset called DBLP-QuAD. The dataset is the largest scholarly KGQA dataset with corresponding SPARQL queries. The dataset contains a wide variety of questions and query types and we present the data generation framework and baseline results. We hope this dataset proves to be a valuable resource for the community.

As future work, we would like to build a robust question answering system for scholarly data using this dataset.

6.11 Acknowledgements

This research was supported by grants from NVIDIA and utilized NVIDIA 2 x RTX A5000 24GB. Furthermore, we acknowledge the financial support from the Federal Ministry for Economic Affairs and Energy of Germany in the project CoyPu (project number 01MK21007[G]) and the German Research Foundation in the project NFDI4DS (project number 460234259). This research is additionally funded by the "Idea and Venture Fund" research grant by Universität Hamburg, which is part of the Excellence Strategy of the Federal and State Governments.

7

DBLPLink: An Entity Linker for the DBLP Scholarly Knowledge Graph

7.1 Bibliographic Information

Debayan Banerjee, Arefa, Ricardo Usbeck and Chris Biemann. 2023. DBLPLink: An Entity Linker for the DBLP Scholarly Knowledge Graph, in the 22nd International Semantic Web Conference. Posters and Demos Track. Athens, Greece.
https://ceur-ws.org/Vol-3632/ISWC2023_paper_428.pdf

7.2 Abstract

In this work, we present a web application named DBLPLink, which performs entity linking over the DBLP scholarly knowledge graph. DBLPLink uses text-to-text pre-trained language models, such as T5, to produce entity label spans from an input text question. Entity candidates are fetched from a database based on the labels, and an entity re-ranker sorts them based on entity embeddings, such as TransE, DistMult and ComplEx. The results are displayed so that users may compare and contrast the results between T5-small, T5-base and the different KG embeddings used. The demo can be accessed at https://ltdemos.informatik.uni-hamburg.de/dblplink/. Code and data shall be made available at https://github.com/uhh-lt/dblplink.

7.3 Introduction and Related Work

Entity Linking (EL) is a natural language processing (NLP) task that involves associating named entities mentioned in text to their corresponding unique identifiers in a knowledge graph (KG). For example, in the question: *Who is the president of USA?*, the named entity span of *USA* has to be linked to the unique identifier `Q30`[1] in the Wikidata KG (Vrandečić and Krötzsch, 2014). Several entity linkers exist (Sevgili et al., 2022) over

[1] https://www.wikidata.org/wiki/Q30

general purpose KGs such as Wikidata, and more specialized KGs, such as bio-medical (French and McInnes, 2023) or financial KGs (Elhammadi et al., 2020), however, to the best of our knowledge, no working entity linker exists for scholarly KGs.

A scholarly KG is a special sub-class of KGs, which contains bibliographic information about research publications, authors, institutions etc. Some well-known scholarly KGs are the OpenAlex[2], ORKG[3] and DBLP[4]. In this work, we focus on the DBLP KG, which caters specifically to computer science, and as a result, is smaller in size than other scholarly KGs. DBLP, which used to stand for Data Bases and Logic Programming[5], was created in 1993 by Michael Ley at the University of Trier, Germany (Ley, 2002). At the time of its release[6], the RDF dump consisted of 2,941,316 person entities, 6,010,605 publication entities, and 252,573,199 RDF triples.

DBLPLink can handle simple and complex questions pertaining to authorship, venues, institutions and other information available in the DBLP KG.

7.4 Web Interface

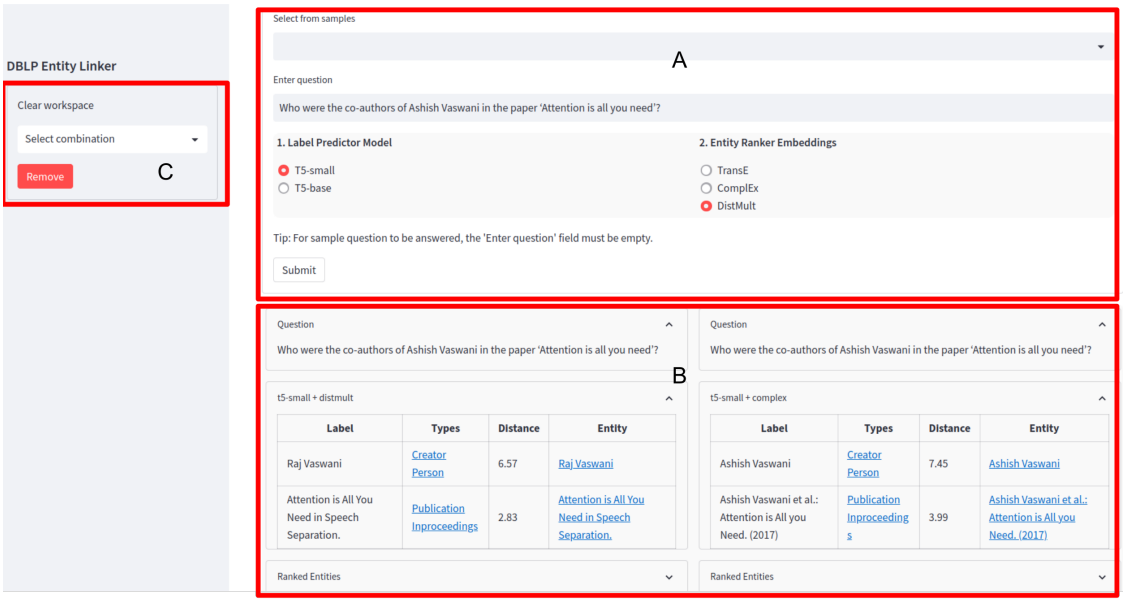


Figure 7.1: User interface of DBLPLink. The question reads: "Who were the co-authors of Ashish Vaswani in the paper 'Attention is all you need'?"

As shown in Figure 7.1, the UI consists of three main parts. In **Section A**, the user can either type a question as input or select a question from the drop-down menu. Further, the user can select which model to use for label span detection, and which embeddings to use for re-ranking of entities. In **Section B**, the results of DBLPLink are displayed. First, the top-ranked entity for each detected span is displayed, with a corresponding label and type from the DBLP KG. A hyperlink to the entity, which points to the original

[2] http://openalex.org/
[3] https://orkg.org/
[4] https://dblp.org/
[5] https://en.wikipedia.org/wiki/DBLP
[6] https://blog.dblp.org/2022/03/02/dblp-in-rdf/

DBLP entity web page is also shown. Additionally, a `distance` metric is shown which denotes how close a match this entity is to the input question. A lower distance means a better match. Towards the bottom of the UI, we can briefly see collapsible boxes called "Ranked Entities", which further display the top 10 ranked entities for each of the detected label spans. Lastly, in **Section C**, the user has an option to remove certain combinations of results from the screen, if the UI becomes too cluttered. Our expectation is that the user shall try multiple combinations of T5 and entity embeddings to compare and contrast the results, which may need occasional cleanup from the UI.

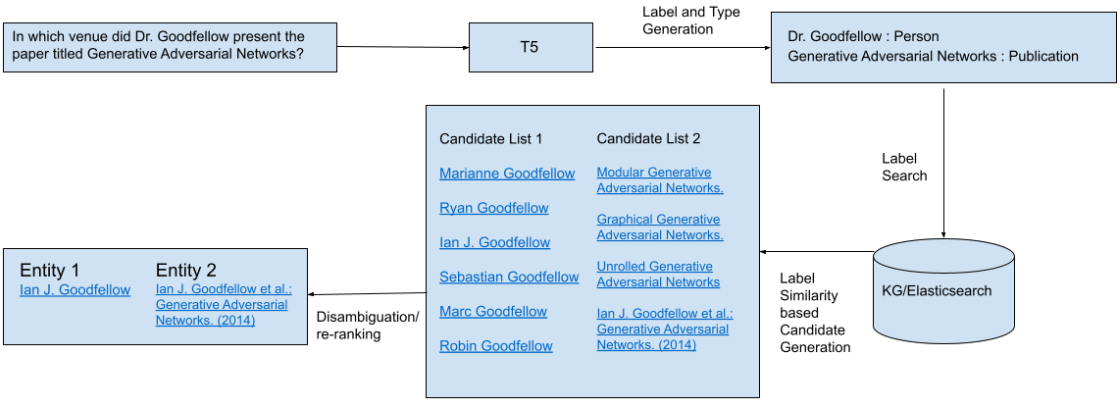


Figure 7.2: Architecture of DBLPLink.

7.5 Architecture

7.5.1 Label and Type Generation

As seen in Figure 7.2, the first step is to produce salient labels and types from the given input question. For this purpose, we use the DBLP-QuAD (Banerjee, Awale, et al., 2023) dataset to fine-tune T5-small and T5-base (Raffel et al., 2020) models, on the task of producing entity labels and types from the input question.

7.5.2 Candidate Generation

With the entity labels and types produced in the previous step, a free-text-search is performed on an Elasticsearch[7] instance, which contains entity URLs with their corresponding labels. The results are further filtered by the types. This gives us a list of candidate entities. In normal operation of the demo application, we present the top-ranked candidate as the final linked entity. We only proceed to the disambiguation stage if the top entity candidate has a label, that is the same as another entity in the candidate list.

7.5.3 Disambiguation

In case two entities in the candidate list share the same label, we proceed with disambiguation, which requires a further re-ranking of the candidate list. For this, we

[7] https://www.elastic.co/

		conditional-disambiguation			hard-disambiguation		
	Label Sort	TransE	ComplEx	DistMult	TransE	ComplEx	DistMult
T5-small	0.698	0.700	0.692	0.699	0.511	0.482	0.537
T5-base	0.698	**0.701**	0.692	**0.701**	0.521	0.484	0.547

Table 7.1: F1-scores for the entity linking task across different combinations of span detector and entity re-ranker

follow a common approach of using Siamese neural networks (Bromley et al., 1993) for learning text similarity between text pairs (Ranasinghe et al., 2019). We embed the input question and the candidate entities in a common embedding space. For this purpose, we create a 969-dimensional embedding, where for a given question, we use the first 768 dimensions for the BERT embedding. We fill the remaining 201 dimensions with zeros. For the entity candidates, we fill the first 768 dimensions with the BERT embedding of the entity label, while the next 200 dimensions are reserved for the entity embeddings. We use three different kinds of embeddings in our experiments, namely TransE (Bordes, Usunier, Garcia-Duran, et al., 2013), ComplEx (Trouillon et al., 2016), and DistMult (Yang et al., 2015). For the remaining 969th dimension, we store the degree of string similarity match between the entity label and the input question. For training, pairs of positive and negative samples are used with a triplet ranking loss function and L2 distance metric.

During inference, a question and an entity candidate are vectorised and passed through the trained Siamese network. The cosine distance between the two resulting embeddings is computed, and the pair with the lowest distance is considered the most suitable match.

7.6 Evaluation

We evaluate our entity linker on the 2.000 questions of the test split of the DBLP-QuAD dataset and measure the F1-score. In Table 7.1, under the heading ‘Label Sorting‘, we consider the top-ranked candidate after the label sorting phase as the linked entity. We perform no further disambiguation. Under the ‘conditional-disambiguation‘ setting, we perform disambiguation only if two entities in the candidate list share the same label. Under the ‘hard-disambiguation‘ setting, re-ranking based on Siamese network cosine distances is always run after the candidate generation phase, essentially ignoring the label sorting order.

We see that hard-disambiguation lags behind significantly in performance when compared to plain label sorting, which points to the learning that for DBLP KG, degree of string match of an author or a publication is more important than the KG embeddings. Based on this finding, we allow the web application to run in ‘conditional-disambiguation‘ mode for better performance. In the case of conditional disambiguation, performance is marginally better when using TransE and DistMult when compared to label sorting, because not many cases of ambiguous labels exist in the DBLP-QuAD test set. However, it is evident from the hard disambiguation case, that DistMult performs the best on a pure disambiguation task. This may be explained by the inherent suitability

of DistMult for 1-to-N relationships, which is close to the nature of the DBLP KG model, where one author may have several papers. On the contrary, TransE expects 1-to-1 relationships, while ComplEx works better for symmetric relationships. Another interesting outcome of the experiments is that the difference in parameter sizes of T5-small and T5-base does not produce any difference in performance. This may be explained by the fact that in the span label production task, much of the focus is on copying the right part of the input to the output. Since the learned knowledge of the model weights from the pre-training task is not being exploited, the larger size of T5-base does not seem to matter.

7.7 Conclusion

In this work, we presented DBLPLink, which is a web-based demonstration of an entity linker over the DBLP scholarly KG. In the future, we would like to add further interactivity to the UI where users can provide feedback on quality of the results. Additionally, a conversational interface for question answering would be desirable for question answering tasks, and we would like to build it in a future version.

7.8 Acknowledgements

This research is performed as a part of the ARDIAS project, funded by the “Idea and Venture Fund“ research grant by Universität Hamburg, which is part of the Excellence Strategy of the Federal and State Governments. This work has additionally received funding through the German Research Foundation (DFG) project NFDI4DS (no. 460234259).

8 Conclusions

8.1 Summary

In this thesis, we have demonstrated that with certain modifications, generative LMs can be effectively used at different stages of the KGQA pipeline. This directly addresses our research questions RQ1 and RQ2.

In our work **Modern Baselines for SPARQL Semantic Parsing** (Banerjee, Nair, Kaur, et al., 2022), we show that when entities and relations are pre-linked, T5 and BART models can be used to generate the logical forms for given questions. For T5, special characters of the SPARQL vocabulary must first be replaced with sentinel tokens, which are special tokens from the T5 tokenizer vocabulary used during the pre-training task. For BART, we discovered a peculiar problem specific to the copying of tokens from the input question to the output logical form. We found that generative LMs are able to mimic the ability of Pointer Generator Networks in copying parts of input tokens to the output logical form, which enables a special class of questions to be answered. On the LC-QuAD 1.0 and 2.0 datasets, T5 models produced state-of-the-art results on the KGQA semantic parsing task.

In our work **GETT-QA: Graph Embedding based T2T Transformer for Knowledge Graph Question Answering** (Banerjee, Nair, Usbeck, et al., 2023a), we demonstrate the role of T5 as an effective label-generator for the entity and relation linking steps. When combined with T5's ability to generate logical forms, this results in an end-to-end KGQA system. Apart from generating entity and relation labels, in this work, we show T5's residual ability to learn simple embedding spaces and use this ability to perform better entity linking for enhanced KGQA performance, attaining the best results on the SimpleQuestions-Wikidata and LC-QuAD 2.0 datasets, when compared to the available KGQA systems.

In our work **The Role of Output Vocabulary in T2T LMs for SPARQL Semantic Parsing** (Banerjee, Nair, Usbeck, et al., 2023b), we discover that T5's ability to generate logical forms is enhanced when the SPARQL vocabulary is replaced with a well-known linguistic vocabulary, e.g., the English language. We perform experiments with variants of this vocabulary, e.g., single characters, multiple characters, and dictionary words.

We evaluate the performance of T5-Small and T5-Base with fine-tuning and prompt-tuning over the GrailQA dataset.

In our work **DBLP-QuAD: A Question Answering Dataset over the DBLP Scholarly Knowledge Graph** (Banerjee, Awale, et al., 2023), we produced a new SPARQL-annotated KGQA dataset for the scholarly domain. This in-domain dataset aims to negate the pre-trained model's advantage of having seen general knowledge during pre-training and is expected to lay down a new benchmark for future researchers in the field of KGQA and semantic parsing. To aid future researchers in solving this dataset, we developed an entity linker for DBLP in Banerjee, Arefa, et al. (2023), which is a working web-based demonstration that can also be accessed over an API.

It must be noted that while we pushed state-of-the-art further with better results on certain datasets in this thesis, the overall performance of models on the task of KGQA is far from perfect. When entities and relations are not pre-linked, accuracy still lies below 50% on complex KGQA tasks, and finding better models to reach human accuracy remains an open challenge.

8.2 Impact

Our work in **Modern Baselines for SPARQL Semantic Parsing** (Banerjee, Nair, Kaur, et al., 2022) is recognized by Gu, Pahuja, et al. (2021), Reyd and Zouaq (2023), Stengel-Eskin and Van Durme (2023) and Hirigoyen et al. (2022), as being one of the first works to explore generative LMs on the KGQA semantic parsing task.

In **GETT-QA: Graph Embedding based T2T Transformer for Knowledge Graph Question Answering** (Banerjee, Nair, Usbeck, et al., 2023a), we discover that a text-to-text model also has a residual capacity to learn a limited embedding space and produce it as output. To the best of our knowledge, this aspect of generative LMs has never been explored before. Moreover, the straightforward end-to-end design is easy to adapt for KGQA on a new domain. For example, Tran et al. (2023) uses our architecture for KGQA in the chemistry domain.

In **The Role of Output Vocabulary in T2T LMs for SPARQL Semantic Parsing** (Banerjee, Nair, Usbeck, et al., 2023b), we highlight that pre-trained LMs perform better when they work with linguistic vocabulary. A similar observation was made subsequently by Lehmann, Gattogi, et al. (2023) on the task of KGQA semantic parsing, where they found that, when SPARQL queries are converted to a controlled natural language format, the models perform better.

In **DBLPLink: An Entity Linker for the DBLP Scholarly Knowledge Graph** (Banerjee, Arefa, et al., 2023), we developed the first public entity linking demo for the DBLP scholarly KG, which was used by some participants of the 1st Scholarly Question Answering Challenge (Banerjee, Usbeck, et al., 2023). The task in this challenge was to solve the DBLP_QuAD (Banerjee, Awale, et al., 2023) dataset, which is a part of this thesis in **DBLP-QuAD: A Question Answering Dataset over the DBLP Scholarly Knowledge Graph**. Seven participating teams employed different models and methods to tackle the challenges of the dataset. Most notably, LLMs were employed in zero-shot and few-shot fashion with the best results for some of the tasks of the challenge, but not all. In the entity linking task, an ASMR tree-based representation produced the best results, which shows that LLMs have limitations when applied to a domain-specific dataset.

8.3 Limitations

In this thesis, we have not experimented with the most recent versions of Large Language Models (LLMs), namely ChatGPT, GPT-3, LLaMA, etc. Our experiments in this thesis are confined to fine-tuning models in the parameter size range of 60M to 220M. Fine-tuning larger models proved to be prohibitively resource-intensive in an academic setting.

In the meantime, two classes of LLMs have appeared which have taken the NLP world by storm. First, are larger LMs that breach the 1B parameter count, e.g., GPT-2 XL with 1.5B parameters, T5-3B, and GPT-3 with 175B parameters. They remain Transformer-based architectures and are still pre-trained in a self-supervised next-token-prediction fashion. However, it has been claimed that at such large model sizes, the appearance of *emergent abilities* (Wei, Tay, et al., 2022) allows these models to perform tasks that were not seen in the training data. This claim has been disputed recently by Lu et al. (2023), who insist that such abilities are merely variants of another ability, called *in-context learning*. In-context learning (also commonly referred to as few-shot approach)(Brown et al., 2020; P Liu et al., 2023) is a setting where a few exemplars of the task to be performed are provided as the input prompt, leading to the model learning the task and performing it on unseen data. This requires no parameter update or fine-tuning of the model.

Second are LLMs, such as ChatGPT, which are *instruction tuned* (Ouyang et al., 2022). Instruction-tuned models rely on reinforcement learning using human feedback (RLHF), or their variants, to align LLMs to better follow human instructions in the input prompt. This kind of tuning reduces the requirement of providing exemplars in the input prompt and is more amenable to conversational use cases between humans and machines. This also leads to creative uses of the model, through human instructions, without the apparent need for parameter updates of the model.

Specifically, ChatGPT was made accessible to the public in November 2022, by which time this thesis had taken shape in terms of published papers, Moreover, the service sits behind a paywall, with the internal details remaining closed to the outside audience. This creates problems from the standpoint of the reproduction of scientific results.

We briefly present the results of other researchers with modern LLMs on the task of KGQA: in Faria et al. (2023), the authors evaluate GPT-3 on the task of KGQA over the QALD-9 dataset. They find that GPT-3 is not suitable for SPARQL generation, both in few-shot and zero-shot settings. Surprisingly, even fine-tuning GPT-3 using the API with 340 sample queries does not exceed the performance of the few-shot setting. However, it is unclear due to the closed setting of GPT-3, how the fine-tuning process is implemented in the back-end. More importantly, when GPT-3 is asked the answer of the same questions directly from within its weights, without generating SPARQL queries, the performance is twice as good. In Klager and Polleres (2023), the authors report a similar lack of performance of GPT-3 and ChatGPT in the KGQA semantic parsing task. Moreover, the authors mention an important difference in the DBpedia and Wikidata KGs. Entities in Wikidata have numeric identifiers, while DBpedia entity identifiers are language-based URIs. They find that DBpedia URIs are easier to generate than Wikidata ones. In general, an LLM can not directly be used for the task of entity linking, due to poor coupling with an external KG. It is common for semantic parsing-based KGQA systems to rely on external entity linkers.

In Xu et al. (2023) and Shu and Z Yu (2023), the authors conclude that fine-tuned models work better than prompted LLMs on the KGQA semantic parsing tasks. This

raises the question of whether LLMs should be used at all for this task since fine-tuning LLMs is prohibitively expensive due to their large parameter count. Moreover, Shu and Z Yu (2023) tries various data augmentation techniques to enable transfer learning of LLMs when working with different KGs, with poor results, raising the issue of poor generalization across KGs.

There is a specific kind of generalization in the domain of semantic parsing, called *compositional generalization*, which remains a highly challenging task. Compositional generalization refers to the ability of a model to generate logical forms for novel combinations of sub-questions from the training set. Unfortunately, LLMs in the default setting do not exhibit any improvement in this field either (Drozdov et al., 2022). However, an advanced prompting technique called *least-to-most prompting* (Zhou et al., 2023), which is related to the concepts of *chain-of-thought prompting* (Wei, Xuezhi Wang, et al., 2023), shows superior performance on this task. In this prompting strategy, tasks are broken down to simpler sub-tasks, and multiple sequential passes are made through the LLM to arrive at the final answer. However, it must be noted that the prompt design, the selection of exemplars, and the decomposition steps vary greatly based on individual datasets.

8.4 Future Work

In the specific case of T5, in our paper **Modern Baselines for SPARQL Semantic Parsing**, we identified issues with the handling of special characters and proposed vocabulary substitution as a means of mitigation of this problem. A possible future approach may be to re-train the T5 tokenizer itself, with special characters of the SPARQL vocabulary included. Additionally, the modern family of LLMs such as LLaMA (Touvron et al., 2023) and Mistral (AQ Jiang et al., 2023) do not suffer from the limitation of dealing with special characters, since they were allowed to remain a part of their pre-training corpus. We recommend the use of these models as a base for future research in semantic parsing.

In our paper **The Role of Output Vocabulary in T2T LMs for SPARQL Semantic Parsing**, we found that replacing SPARQL vocabulary with a linguistic vocabulary improves performance. One hypothesis is that generative LMs perform better with words that were found more often in the pre-training corpus. However, a more detailed analysis of which subset of the linguistic vocabulary (e.g., which group of English words) works best remains to be explored in future work.

In our paper **GETT-QA: Graph Embedding Based T2T Transformer for Knowledge Graph Question Answering**, we explored the ability of a generative LM, apart from generating logical forms, in producing a small embedding space as a string. In reality, an embedding space is inherently numerical in nature, and a generative LM is not the ideal candidate to learn and generate such spaces. We foresee the development of better joint architectures, where the embedding space is learned and generated as the output of a fully connected last layer as numeric values instead of strings.

In our paper **DBLP-QuAD: A Question Answering Dataset over the DBLP Scholarly Knowledge Graph**, we presented a new dataset for the KGQA semantic parsing task on the DBLP bibliographic KG (Ley, 2002). Recently, the SemOpenAlex (Färber et al., 2023) project has made a much larger bibliography available in the form of a KG. Porting of the DBLP_QuAD dataset to SemOpenAlex is a possible direction for future work. Additionally, a more challenging setting exists when retrieving an answer from both

KG and text sources. This form of QA is called *hybrid QA*, and remains to be explored as a research direction in the KGQA world.

Apart from specific cases of future work as discussed above, in the general direction of future KGQA research, much work remains, especially when viewed through the lens of LLMs. While LLMs show outstanding performance in text generation tasks, they also *hallucinate* facts. Since KGs contain gold facts, which are generally human-curated, it is desirable to ground the answers generated by LLMs to KGs. Since KGs sit outside the LLM, it remains an open question on how to integrate the two. In this pursuit, Pan et al. (2023) outlines several possible avenues that researchers may take in the future.

One family of approaches lies in augmenting the pre-training corpus with KG triples (Kang et al., 2022; Xiong et al., 2019). Another attempts to ground LLM answers in the KG during inferencing (P Lewis et al., 2020; Wu et al., 2022). More recently, changes in the attention mechanism in Transformers have been proposed in Bertsch et al. (2023), so that the decoder may directly attend to an external encoder database during inference. A fundamentally different conversation has also started around modifying KGs themselves to be more aligned to the LLM space, by replacing current identifiers with ones in the "word space". For example, in Zhao et al. (2023), the authors show that encoding a sub-graph in text is almost as good as Graph Neural Network generated embeddings, with the added benefit of explainability of word-based representations. In Fatemi et al. (2023), the authors show that appropriately naming KG nodes in word-space, makes it easier for pre-trained models to address them later during inference.

Another possible approach, which has not yet been seen in current literature, is to embed KG triples as tokens in the LLM vocabulary mixed with the normal LLM vocabulary. If a KG triple is represented as a single LLM token, the production of this triple during LLM inference can not be corrupted at an atomic level, where an atom is a triple. This makes the production of triples from LLMs hallucination-free. As a pre-training task in this setup, one may imagine *fact grounding*. The LLM is given a corpus of text, where each textual statement is accompanied by the corresponding triple. The task of the LLM would be to produce the missing tokens when random tokens are masked in this corpus. One issue with this approach is the size of vocabulary when compared to the number of triples in a large KG. Traditionally, LLM vocabularies are several orders of magnitude smaller in size than the number of triples present in a KG. However, the most recent LLMs[1] have started experimenting with larger vocabulary sizes, e.g., 256k instead of 32k. Hence, there is a possibility that this approach may be feasible in the near future.

In the end, we address a question about the relevance of KGs in today's world. LLMs often produce strong performance on QA tasks directly, without the need to refer to a KG, or generate a query. This leads to the question: do we need KGs, or semantic parsing, anymore? While there may be varying answers to this question, one overlooked aspect is that of domain-specific KGs, for example, industrial or internal KGs. LLMs are pre-trained on general information, while custom KGs contain out-of-distribution information in comparison. Generating sufficient and appropriate training data out of custom KGs for pre-training and instruction tuning LLMs is a prohibitively expensive and lengthy process. In support of semantic parsing, or the generation of logical queries, one strong argument is that of *explanability*. Once we have a logical form that can be executed to retrieve answers from a KG, a deeper inspection of the logical form produces a logical explanation of why a particular answer was fetched. If the answer

[1] https://storage.googleapis.com/deepmind-media/gemma/gemma-report.pdf

was incorrect, which part of the logical form produced this inaccuracy can be identified and corrected. In contrast, the LLM behaves like a black box, the internal behavior of which can only be coarsely controlled through prompts. Based on these limitations of LLMs, it is our belief that KGs continue to remain relevant today, and further work is required in the quest to integrate KGs and LLMs in a better and efficient manner, so that the needs of all sub-domains of KGQA can be suitably addressed.

References

Abdalghani Abujabal, Mohamed Yahya, Mirek Riedewald, and Gerhard Weikum (2017). *Automated Template Generation for Question Answering over Knowledge Graphs*. In: *Proceedings of the 26th International Conference on World Wide Web*. Perth, Australia, pp. 1191–1200 (cit. on pp. 4, 41).

David Alvarez-Melis and Tommi S. Jaakkola (2017). *Tree-Structured Decoding with Doubly-Recurrent Neural Networks*. In: *International Conference on Learning Representations*. Poster (cit. on p. 4).

Konstantine Arkoudas, Nicolas Guenon des Mesnards, Melanie Rubino, Sandesh Swamy, Saarthak Khanna, Weiqi Sun, and Khan Haidar (2022). "PIZZA: A New Benchmark for Complex End-to-End Task-Oriented Parsing". In: arxiv.org eprint (cit. on p. 66).

Yoav Artzi and Luke Zettlemoyer (2013). "Weakly Supervised Learning of Semantic Parsers for Mapping Instructions to Actions". In: *Transactions of the Association for Computational Linguistics* 1. Ed. by Dekang Lin and Michael Collins, pp. 49–62 (cit. on p. 4).

Ram G. Athreya, Srividya Kona Bansal, Axel-Cyrille Ngonga Ngomo, and Ricardo Usbeck (2021). *Template-based Question Answering using Recursive Neural Networks*. In: *15th IEEE International Conference on Semantic Computing*. Laguna Hills, CA, USA, pp. 195–198 (cit. on p. 41).

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio (n.d.). *Neural Machine Translation by Jointly Learning to Align and Translate*. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, Conference Track Proceedings* (cit. on p. 22).

Krisztian Balog and Tom Kenter (2019). *Personal Knowledge Graphs: A Research Agenda*. In: *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval*. Santa Clara, CA, USA, pp. 217–220 (cit. on p. 2).

Debayan Banerjee (2023). *Semantic Parsing for Knowledge Graph Question Answering with Large Language Models*. In: *The Semantic Web: ESWC 2023 Satellite Events*. Hersonissos, Greece, pp. 234–243 (cit. on p. 10).

Debayan Banerjee, Arefa, Ricardo Usbeck, and Chris Biemann (2023). *DBLPLink: An Entity Linker for the DBLP Scholarly Knowledge Graph*. In: *Proceedings of the 22nd International Semantic Web Conference Posters, Demos and Industry Tracks*. Vol. 3632. Athens, Greece (cit. on pp. 7, 9 sq., 92).

Debayan Banerjee, Sushil Awale, Ricardo Usbeck, and Chris Biemann (2023). *DBLP-QuAD: A Question Answering Dataset over the DBLP Scholarly Knowledge Graph*. In: *Proceedings of the 13th International Workshop on Bibliometric-enhanced Information Retrieval (BIR)*. Dublin, Ireland, pp. 37–51 (cit. on pp. 7, 9, 88, 92).

Debayan Banerjee, Pranav Ajit Nair, Jivat Neet Kaur, Ricardo Usbeck, and Chris Biemann (2022). *Modern Baselines for SPARQL Semantic Parsing*. In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. Madrid, Spain, pp. 2260–2265 (cit. on pp. 7 sqq., 12, 23, 50, 59, 65, 83, 91 sq.).

Debayan Banerjee, Pranav Ajit Nair, Ricardo Usbeck, and Chris Biemann (2023a). *GETT-QA: Graph Embedding Based T2T Transformer for Knowledge Graph Question Answering*. In: *The*

Semantic Web: 20th International Conference, ESWC 2023, Hersonissos, Crete, Greece, May 28–June 1, 2023, Proceedings, pp. 279–297 (cit. on pp. 7, 9, 91 sq.).

Debayan Banerjee, Pranav Ajit Nair, Ricardo Usbeck, and Chris Biemann (2023b). *The Role of Output Vocabulary in T2T LMs for SPARQL Semantic Parsing*. In: *Findings of the Association for Computational Linguistics: ACL 2023*. Toronto, Canada, pp. 12219–12228 (cit. on pp. 7, 9, 12, 91 sq.).

Debayan Banerjee, Ricardo Usbeck, Nandana Mihindukulasooriya, Mohamad Yaser Jaradeh, Sören Auer, Gunjan Singh, Raghava Mutharaju, and Pavan Kapanipathi, eds. (2023). *Joint Proceedings of Scholarly QALD 2023 and SemREC 2023 co-located with 22nd International Semantic Web Conference*. 3592. Athens, Greece (cit. on p. 92).

Yehoshua Bar-Hillel (1953). "A Quasi-Arithmetical Notation for Syntactic Description". In: *Language* 29.1, pp. 47–58 (cit. on p. 4).

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang (2013a). *Semantic Parsing on Freebase from Question-Answer Pairs*. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA, pp. 1533–1544 (cit. on p. 4).

— (2013b). *Semantic Parsing on Freebase from Question-Answer Pairs*. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA, pp. 1533–1544 (cit. on p. 75).

Jonathan Berant and Percy Liang (2014). *Semantic Parsing via Paraphrasing*. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Baltimore, Maryland, pp. 1415–1425 (cit. on p. 4).

Amanda Bertsch, Uri Alon, Graham Neubig, and Matthew R. Gormley (2023). *Unlimiformer: Long-Range Transformers with Unlimited Length Input*. In: *Conference on Neural Information Processing Systems (NeurIPS)*. Virtual Poster. New Orleans, USA (cit. on p. 95).

Nikita Bhutani, Xinyi Zheng, and H. V. Jagadish (2019). *Learning to Answer Complex Questions over Knowledge Bases with Query Composition*. In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. Beijing, China, pp. 739–748 (cit. on pp. 4, 51).

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor (2008). *Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge*. In: *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. Vancouver, Canada, pp. 1247–1250 (cit. on pp. 1, 41, 68, 73).

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston (2015). "Large-scale Simple Question Answering with Memory Networks". In: arxiv.org eprint (cit. on pp. 56, 58).

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko (2013). *Translating Embeddings for Modeling Multi-relational Data*. In: *27th Annual Conference on Neural Information Processing Systems*, pp. 2787–2795 (cit. on pp. 36, 45, 53, 89).

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah (1993). *Signature Verification Using a "Siamese" Time Delay Neural Network*. In: *Proceedings of the 6th International Conference on Neural Information Processing Systems*. Denver, Colorado, pp. 737–744 (cit. on p. 89).

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei (2020). *Language Models are Few-Shot Learners*. In: *Advances in Neural Information Processing Systems*. Vol. 33. Virtual Event, pp. 1877–1901 (cit. on pp. 65, 93).

Qingqing Cai and Alexander Yates (2013). *Large-scale Semantic Parsing via Schema Matching and Lexicon Extension*. In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Sofia, Bulgaria, pp. 423–433 (cit. on p. 75).

Shulin Cao, Jiaxin Shi, Liangming Pan, Lunyiu Nie, Yutong Xiang, Lei Hou, Juanzi Li, Bin He, and Hanwang Zhang (2022). *KQA Pro: A Dataset with Explicit Compositional Programs for Complex Question Answering over Knowledge Base*. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pp. 6101–6119 (cit. on p. 76).

Nilesh Chakraborty, Denis Lukovnikov, Gaurav Maheshwari, Priyansh Trivedi, Jens Lehmann, and Asja Fischer (2019). *Introduction to Neural Network based Approaches for Question Answering over Knowledge Graphs*. arxiv.org eprint (cit. on p. 74).

Angel X. Chang and Christopher Manning (2012). *SUTime: A Library for Recognizing and Normalizing Time Expressions*. In: *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*. Istanbul, Turkey, pp. 3735–3740 (cit. on p. 41).

Viktoriia Chekalina, Anton Razzhigaev, Albert Sayapin, Evgeny Frolov, and Alexander Panchenko (2022). *MEKER: Memory Efficient Knowledge Embedding Representation for Link Prediction and Question Answering*. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*. Dublin, Ireland, pp. 355–365 (cit. on pp. 51, 57).

Xilun Chen, Asish Ghoshal, Yashar Mehdad, Luke Zettlemoyer, and Sonal Gupta (2020). *Low-Resource Domain Adaptation for Compositional Task-Oriented Semantic Parsing*. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Online, pp. 5090–5100 (cit. on p. 66).

Yongrui Chen, Huiying Li, Yuncheng Hua, and Guilin Qi (2020). *Formal Query Building with Query Structure Prediction for Complex Question Answering over Knowledge Base*. In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*. Yokohama, Japan, pp. 3751–3758 (cit. on pp. 41, 44).

Jianpeng Cheng and Mirella Lapata (2018). *Weakly-Supervised Neural Semantic Parsing with a Generative Ranker*. In: *Proceedings of the 22nd Conference on Computational Natural Language Learning*. Brussels, Belgium: Association for Computational Linguistics, pp. 356–367 (cit. on p. 4).

Jianpeng Cheng, Siva Reddy, Vijay Saraswat, and Mirella Lapata (2017). *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)*. Vancouver, Canada, pp. 44–55 (cit. on p. 4).

— (2019). "Learning an Executable Neural Semantic Parser". In: *Computational Linguistics* 45.1, pp. 59–94 (cit. on p. 4).

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio (2014). *Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation*. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar, pp. 1724–1734 (cit. on pp. 4, 19, 21).

Philipp Christmann, Rishiraj Saha Roy, Abdalghani Abujabal, Jyotsna Singh, and Gerhard Weikum (2019). *Look before you Hop: Conversational Question Answering over Knowledge Graphs Using Judicious Context Expansion*. In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. Beijing, China, pp. 729–738 (cit. on p. 51).

Hejie Cui, Jiaying Lu, Shiyu Wang, Ran Xu, Wenjing Ma, Shaojun Yu, Yue Yu, Xuan Kan, Chen Ling, Liang Zhao, Joyce Ho, Fei Wang, and Carl Yang (2023). "A Survey on Knowledge Graphs for Healthcare: Resources, Applications, and Promises". In: arxiv.org eprint (cit. on p. 2).

Rajarshi Das, Manzil Zaheer, Dung Thai, Ameya Godbole, Ethan Perez, Jay Yoon Lee, Lizhen Tan, Lazaros Polymenakos, and Andrew McCallum (2021). *Case-based Reasoning for Natural Language Queries over Knowledge Bases*. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic, pp. 9594–9611 (cit. on p. 4).

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota, USA, pp. 4171–4186 (cit. on pp. 5, 30, 65).

Dennis Diefenbach, Andreas Both, Kamal Deep Singh, and Pierre Maret (2020). "Towards a Question Answering System over the Semantic Web". In: *Semantic Web – Interoperability, Usability, Applicability* 11.3, pp. 421–439 (cit. on p. 51).

Dennis Diefenbach, Vanessa Lopez, Kamal Singh, and Pierre Maret (2017). "Core Techniques of Question Answering Systems over Knowledge Bases: a Survey". In: *Knowledge and Information Systems*, pp. 529–569 (cit. on p. 40).

Dennis Diefenbach, Thomas Pellissier Tanon, Kamal Deep Singh, and Pierre Maret (2017). *Question Answering Benchmarks for Wikidata*. In: *Proceedings of the ISWC Posters & Demonstrations and Industry Tracks co-located with 16th International Semantic Web Conference*. Vienna, Austria (cit. on p. 56).

Jiwei Ding, Wei Hu, Qixin Xu, and Yuzhong Qu (2019). *Leveraging Frequent Query Substructures to Generate Formal Queries for Complex Question Answering*. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China, pp. 2614–2622 (cit. on pp. 41, 44 sqq.).

Ning Ding, Shengding Hu, Weilin Zhao, Yulin Chen, Zhiyuan Liu, Haitao Zheng, and Maosong Sun (2022). *OpenPrompt: An Open-source Framework for Prompt-learning*. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*. Dublin, Ireland, pp. 105–113 (cit. on p. 67).

Daniel Diomedi and Aidan Hogan (2021). "Question Answering over Knowledge Graphs with Neural Machine Translation and Entity Linking". In: arxiv.org eprint (cit. on pp. 4, 51, 58).

Li Dong and Mirella Lapata (2016). *Language to Logical Form with Neural Attention*. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany, pp. 33–43 (cit. on p. 4).

Andrew Drozdov, Nathanael Schärli, Ekin Akyürek, Nathan Scales, Xinying Song, Xinyun Chen, Olivier Bousquet, and Denny Zhou (2022). "Compositional Semantic Parsing with Large Language Models". In: arxiv.org eprint (cit. on p. 94).

Mohnish Dubey, Debayan Banerjee, Abdelrahman Abdelkawi, and Jens Lehmann (2019). *LC-QuAD 2.0: A Large Dataset for Complex Question Answering over Wikidata and DBpedia*. In: *The Semantic Web – ISWC*. Auckland, New Zealand, pp. 69–78 (cit. on pp. 40, 55, 75).

Mohnish Dubey, Debayan Banerjee, Debanjan Chaudhuri, and Jens Lehmann (2018). *EARL: Joint Entity and Relation Linking for Question Answering over Knowledge Graphs*. In: *The Semantic Web – ISWC 2018*. Monterey, CA, USA, pp. 108–126 (cit. on p. 41).

Mohnish Dubey, Sourish Dasgupta, Ankit Sharma, Konrad Höffner, and Jens Lehmann (2016). *AskNow: A Framework for Natural Language Query Formalization in SPARQL*. In: *Proceedings of the 13th International Conference on The Semantic Web. Latest Advances and New Domains - Volume 9678*. Berlin, Heidelberg, pp. 300–316 (cit. on p. 74).

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith (2015). *Transition-Based Dependency Parsing with Stack Long Short-Term Memory*. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th*

International Joint Conference on Natural Language Processing (Volume 1: Long Papers). Beijing, China, pp. 334–343 (cit. on p. 5).

Sarah Elhammadi, Laks V.S. Lakshmanan, Raymond Ng, Michael Simpson, Baoxing Huai, Zhefeng Wang, and Lanjun Wang (2020). *A High Precision Pipeline for Financial Knowledge Graph Construction*. In: *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online), pp. 967–977 (cit. on p. 87).

Keyur Faldu, Amit P. Sheth, Prashant Kikani, and Hemang Akabari (2021). "KI-BERT: Infusing Knowledge Context for Better Language and Domain Understanding". In: arxiv.org eprint (cit. on p. 51).

Michael Färber, David Lamprecht, Johan Krause, Linn Aung, and Peter Haase (2023). *SemOpenAlex: The Scientific Landscape in 26 Billion RDF Triples*. In: *The Semantic Web , Part II*. Athens, Greece, pp. 94–112 (cit. on pp. 2, 94).

Bruno Faria, Dylan Perdigão, and Hugo Gonçalo Oliveira (2023). *Question Answering over Linked Data with GPT-3*. In: *12th Symposium on Languages, Applications and Technologies*. Vol. 113. Dublin, Ireland, 1:1–1:15 (cit. on p. 93).

Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi (2023). "Talk like a Graph: Encoding Graphs for Large Language Models". In: arxiv.org eprint (cit. on p. 95).

J. R. Firth (1957). "A Synopsis of Linguistic Theory 1930-55." In: 1952-59, pp. 1–32 (cit. on p. 13).

Evan French and Bridget T. McInnes (2023). "An Overview of Biomedical Entity Linking Throughout the Years". In: *Journal of Biomedical Informatics* 137, pp. 104–252 (cit. on p. 87).

Daniel Furrer, Marc van Zee, Nathan Scales, and Nathanael Schärli (2020). *Compositional Generalization in Semantic Parsing: Pre-training vs. Specialized Architectures*. arxiv.org eprint (cit. on p. 41).

Mikhail Galkin, Priyansh Trivedi, Gaurav Maheshwari, Ricardo Usbeck, and Jens Lehmann (2020). *Message Passing for Hyper-Relational Knowledge Graphs*. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Online, pp. 7346–7359 (cit. on pp. 45, 56).

Aaron Gokaslan and Vanya Cohen (2019). *OpenWebText Corpus* (cit. on p. 43).

Alex Graves (2013). "Generating Sequences With Recurrent Neural Networks". In: arxiv.org eprint (cit. on p. 22).

Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su (2021). *Beyond I.I.D.: Three Levels of Generalization for Question Answering on Knowledge Bases*. In: *Proceedings of the Web Conference*. Ljubljana, Slovenia, pp. 3477–3488 (cit. on pp. 4 sq., 66, 75, 80).

Yu Gu, Vardaan Pahuja, Gong Cheng, and Yu Su (2021). *Knowledge Base Question Answering: A Semantic Parsing Perspective*. In: *4th Conference on Automated Knowledge Base Construction, AKBC*. Online and London, UK (cit. on p. 92).

Claudio Gutierrez and Juan F. Sequeda (2021). "Knowledge Graphs". In: *Communincations of ACM* 64.3, pp. 96–104 (cit. on p. 2).

William L. Hamilton (n.d.). "Graph Representation Learning". In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 14.3 (), pp. 1–159 (cit. on p. 38).

Zellig S. Harris (1954). "Distributional Structure". In: *WORD* 10.2-3, pp. 146–162 (cit. on p. 13).

Bin He, Di Zhou, Jinghui Xiao, Xin Jiang, Qun Liu, Nicholas Jing Yuan, and Tong Xu (2020). *BERT-MK: Integrating Graph Contextualized Knowledge into Pre-trained Language Models*. In: *Findings of the Association for Computational Linguistics: EMNLP*. Online, pp. 2281–2290 (cit. on p. 51).

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). *Deep Residual Learning for Image Recognition*. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778 (cit. on p. 28).

Jonathan Herzig, Peter Shaw, Ming-Wei Chang, Kelvin Guu, Panupong Pasupat, and Yuan Zhang (2021). "Unlocking Compositional Generalization in Pre-trained Models Using Intermediate Representations". In: arxiv.org eprint (cit. on p. 41).

Rose Hirigoyen, Amal Zouaq, and Samuel Reyd (2022). *A Copy Mechanism for Handling Knowledge Base Elements in SPARQL Neural Machine Translation*. In: *Findings of the Association for Computational Linguistics: AACL-IJCNLP*. Online only, pp. 226–236 (cit. on p. 92).

Frank L. Hitchcock (1927). "The Expression of a Tensor or a Polyadic as a Sum of Products". In: *Journal of Mathematics and Physics* 6.1-4, pp. 164–189 (cit. on p. 51).

Sepp Hochreiter and Jürgen Schmidhuber (1997). "Long Short-Term Memory". In: *Neural Computation* 9.8, pp. 1735–1780 (cit. on pp. 4, 12, 19 sq., 44).

Konrad Höffner, Sebastian Walter, Edgard Marx, Ricardo Usbeck, Jens Lehmann, and Axel-Cyrille Ngonga Ngomo (2017). "Survey on Challenges of Question Answering in the Semantic Web". In: *Semantic Web* 8.6, pp. 895–920 (cit. on p. 2).

Sen Hu, Lei Zou, and Xinbo Zhang (2018). *A State-transition Framework to Answer Complex Questions over Knowledge Base*. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium, pp. 2098–2108 (cit. on p. 41).

Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li (2019). *Knowledge Graph Embedding Based Question Answering*. In: *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. Melbourne VIC, Australia, pp. 105–113 (cit. on p. 51).

Mohamad Yaser Jaradeh, Markus Stocker, and Sören Auer (2020). *Question Answering on Scholarly Knowledge Graphs*. In: *Digital Libraries for Open Knowledge*, pp. 19–32 (cit. on p. 74).

Zhen Jia, Soumajit Pramanik, Rishiraj Saha Roy, and Gerhard Weikum (2021). *Complex Temporal Question Answering on Knowledge Graphs*. In: *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*. Virtual Event, Queensland, Australia, pp. 792–802 (cit. on pp. 41, 51).

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed (2023). *Mistral 7B*. arxiv.org eprint (cit. on p. 94).

Martin Joos (1950). "Description of Language Design". In: *Journal of the Acoustical Society of America* 22, pp. 701–707 (cit. on p. 13).

Dan Jurafsky and James H. Martin (2009). *Speech and Language Processing : An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition* (cit. on pp. 13, 15 sq., 18 sq., 21 sq., 26, 29, 31, 33 sq.).

Endri Kacupaj, Joan Plepi, Kuldeep Singh, Harsh Thakkar, Jens Lehmann, and Maria Maleshkova (2021). *Conversational Question Answering over Knowledge Graphs with Transformer and Graph Attention Networks*. In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Online, pp. 850–862 (cit. on p. 52).

Nal Kalchbrenner and Phil Blunsom (2013). *Recurrent Continuous Translation Models*. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA, pp. 1700–1709 (cit. on p. 21).

Aishwarya Kamath and Rajarshi Das (2018). "A Survey on Semantic Parsing". In: arxiv.org eprint (cit. on p. 4).

Minki Kang, Jinheon Baek, and Sung Ju Hwang (2022). *KALA: Knowledge-Augmented Language Model Adaptation*. In: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Seattle, United States, pp. 5144–5167 (cit. on p. 95).

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet (2020). *Measuring Compositional Generalization: A Comprehensive Method on Realistic Data*. In: *International Conference on Learning Representations*. Virtual Poster (cit. on pp. 41, 76, 84).

Diederik P. Kingma and Jimmy Ba (2015). *Adam: A Method for Stochastic Optimization*. In: *3rd International Conference on Learning Representations*. Poster. San Diego, CA, USA (cit. on p. 62).

Gerhard Georg Klager and Axel Polleres (2023). *Is GPT fit for KGQA?* In: *Proceedings of the International Workshop on Knowledge Graph Generation from Text, co-located with Extended Semantic Web Conference 2023*. Hersonissos, Greece (cit. on p. 93).

Taku Kudo and John Richardson (n.d.). *SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing*. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Brussels, Belgium, pp. 66–71 (cit. on p. 29).

Pawan Kumar and Srikanta Bedathur (2020). "A Survey on Semantic Parsing from the perspective of Compositionality". In: arxiv.org eprint (cit. on p. 4).

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. (2019). "Natural Questions: A Benchmark for Question Answering Research". In: *Transactions of the Association for Computational Linguistics* 7, pp. 453–466 (cit. on p. 74).

Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen (2021). *A Survey on Complex Knowledge Base Question Answering: Methods, Challenges and Solutions*. In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pp. 4483–4491 (cit. on pp. 2, 50).

Yunshi Lan and Jing Jiang (2020). *Query Graph Generation for Answering Multi-hop Complex Questions from Knowledge Bases*. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online, pp. 969–974 (cit. on pp. 41, 44).

Jens Lehmann, Preetam Gattogi, Dhananjay Bhandiwad, Sébastien Ferré, and Sahar Vahdati (2023). *Language Models as Controlled Natural Language Semantic Parsers for Knowledge Graph Question Answering*. In: *26th European Conference on Artificial Intelligence*. Vol. 372. Krakow (Cracovie), Poland, pp. 1348–1356 (cit. on p. 92).

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer (2015). "DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia". In: *Semantic Web Journal* 6.2, pp. 167–195 (cit. on pp. 1, 40, 73).

Adam Lerer, Ledell Wu, Jiajun Shen, Timothee Lacroix, Luca Wehrstedt, Abhijit Bose, and Alex Peysakhovich (2019). *Pytorch-BigGraph: A Large Scale Graph Embedding System*. In: *Proceedings of Machine Learning and Systems*. Vol. 1, pp. 120–131 (cit. on pp. 45, 53).

Brian Lester, Rami Al-Rfou, and Noah Constant (2021). *The Power of Scale for Parameter-Efficient Prompt Tuning*. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic, pp. 3045–3059 (cit. on p. 65).

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer (2020). *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension*. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online, pp. 7871–7880 (cit. on pp. 2, 5, 12 sq., 35 sq., 40).

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela (2020). *Retrieval-Augmented Generation for*

Knowledge-Intensive NLP Tasks. In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. Vancouver, BC, Canada, pp. 9459–9474 (cit. on p. 95).

Michael Ley (2002). "The DBLP Computer Science Bibliography: Evolution, Research Issues, Perspectives". In: *String Processing and Information Retrieval*. Vol. 2476, pp. 1–10 (cit. on pp. 7, 76, 87, 94).

Xiang Lisa Li and Percy Liang (2021). *Prefix-Tuning: Optimizing Continuous Prompts for Generation*. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online, pp. 4582–4597 (cit. on pp. 65, 67).

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig (2023). "Pre-Train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing". In: *ACM Computing Surveys* 55.9 (cit. on p. 93).

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang (2020). *K-BERT: Enabling Language Representation with Knowledge Graph*. In: vol. 34. 03. New York, USA, pp. 2901–2908 (cit. on p. 51).

Vanessa Lopez and Enrico Motta (2004). *Ontology-Driven Question Answering in AquaLog*. In: *Natural Language Processing and Information Systems*. Berlin, Heidelberg, pp. 89–102 (cit. on p. 3).

Vanessa Lopez, Enrico Motta, and Victoria Uren (2006). *PowerAqua: Fishing the Semantic Web*. In: *The Semantic Web: Research and Applications*. Berlin, Heidelberg, pp. 393–410 (cit. on p. 3).

Ilya Loshchilov and Frank Hutter (2019). *Decoupled Weight Decay Regularization*. In: *7th International Conference on Learning Representations*. New Orleans, LA, USA (cit. on p. 67).

Sheng Lu, Irina Bigoulaeva, Rachneet Sachdeva, Harish Tayyar Madabushi, and Iryna Gurevych (2023). "Are Emergent Abilities in Large Language Models just In-Context Learning?" In: arxiv.org eprint (cit. on p. 93).

Denis Lukovnikov and Asja Fischer (2021). *Insertion-based Tree Decoding*. In: *Findings of the Association for Computational Linguistics: ACL-IJCNLP*. Online, pp. 3201–3213 (cit. on p. 5).

Denis Lukovnikov, Asja Fischer, Jens Lehmann, and Sören Auer (2017). *Neural Network-Based Question Answering over Knowledge Graphs on Word and Character Level*. In: *Proceedings of the 26th International Conference on World Wide Web*. Perth, Australia, pp. 1211–1220 (cit. on p. 3).

Kangqi Luo, Fengli Lin, Xusheng Luo, and Kenny Zhu (2018). *Knowledge Base Question Answering via Encoding of Complex Query Graphs*. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium, pp. 2185–2194 (cit. on pp. 41, 44).

Fabiano Ferreira Luz and Marcelo Finger (2018). "Semantic Parsing Natural Language into SPARQL: Improving Target Language Representation with Neural Attention". In: arxiv.org eprint (cit. on p. 4).

Fang Ma, Chen Zhang, Lei Ren, Jingang Wang, Qifan Wang, Wei Wu, Xiaojun Quan, and Dawei Song (2022). "XPrompt: Exploring the Extreme of Prompt Tuning". In: arxiv.org eprint (cit. on p. 65).

Joel Mackenzie, Rodger Benham, Matthias Petri, Johanne R. Trippas, J. Shane Culpepper, and Alistair Moffat (2020). *CC-News-En: A Large English News Corpus*. In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. Virtual Event, Ireland, pp. 3077–3084 (cit. on pp. 35, 43).

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur (2010). *Recurrent Neural Network Based Language Model*. In: *INTERSPEECH*. Makuhari, Chiba, Japan, pp. 1045–1048 (cit. on p. 17).

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean (2013). *Distributed Representations of Words and Phrases and their Compositionality*. In: *Proceedings of the 26th*

International Conference on Neural Information Processing Systems - Volume 2. Lake Tahoe, Nevada, pp. 3111–3119 (cit. on p. 15).

George A. Miller (1995). "WordNet: A Lexical Database for English". In: *Communications of ACM* 38.11, pp. 39–41 (cit. on p. 3).

Cedric Möller, Jens Lehmann, and Ricardo Usbeck (2022). "Survey on English Entity Linking on Wikidata: Datasets and Approaches". In: *Semantic Web* 13.6, pp. 925–966 (cit. on p. 2).

Sumit Neelam, Udit Sharma, Hima Karanam, Shajith Ikbal, Pavan Kapanipathi, Ibrahim Abdelaziz, Nandana Mihindukulasooriya, Young-Suk Lee, Santosh Srivastava, Cezar Pendus, Saswati Dana, Dinesh Garg, Achille Fokoue, G P Shrivatsa Bhargav, Dinesh Khandelwal, Srinivas Ravishankar, Sairam Gurajada, Maria Chang, Rosario Uceda-Sosa, Salim Roukos, Alexander Gray, Guilherme Lima, Ryan Riegel, Francois Luus, and L V Subramaniam (2022). *SYGMA: A System for Generalizable and Modular Question Answering Over Knowledge Bases*. In: *Findings of the Association for Computational Linguistics: EMNLP*. Abu Dhabi, United Arab Emirates, pp. 3866–3879 (cit. on p. 51).

Barlas Oguz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Scott Yih (2022). *UniK-QA: Unified Representations of Structured and Unstructured Knowledge for Open-Domain Question Answering*. In: *Findings of the Association for Computational Linguistics: NAACL 2022*. Seattle, United States, pp. 1535–1546 (cit. on p. 51).

C.E. Osgood, G.J. Suci, and P.H. Tenenbaum (1957). *The Measurement of meaning*. University of Illinois Press (cit. on pp. 13 sq.).

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe (2022). *Training Language Models to Follow Instructions with Human Feedback*. In: *Advances in Neural Information Processing Systems*. Vol. 35, pp. 27730–27744 (cit. on p. 93).

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu (2023). "Unifying Large Language Models and Knowledge Graphs: A Roadmap". In: arxiv.org eprint (cit. on p. 95).

Aleksandr Perevalov, Xi Yan, Liubov Kovriguina, Longquan Jiang, Andreas Both, and Ricardo Usbeck (2022). *Knowledge Graph Question Answering Leaderboard: A Community Resource to Prevent a Replication Crisis*. In: *Proceedings of the Thirteenth Language Resources and Evaluation Conference*. Marseille, France, pp. 2998–3007 (cit. on pp. 46, 74).

Joan Plepi, Endri Kacupaj, Kuldeep Singh, Harsh Thakkar, and Jens Lehmann (2021). *Context Transformer with Stacked Pointer Networks for Conversational Question Answering over Knowledge Graphs*. In: *The Semantic Web - ESWC*. Hersonisson, Greece, pp. 356–371 (cit. on p. 52).

Soumajit Pramanik, Jesujoba Alabi, Rishiraj Saha Roy, and Gerhard Weikum (2021). "UNIQORN: Unified Question Answering over RDF Knowledge Graphs and Natural Language Text". In: arxiv.org eprint (cit. on pp. 51, 56).

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever (2018). "Improving Language Understanding by Generative Pre-Training". In: (cit. on p. 2).

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu (2020). "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *Journal of Machine Learning Research* 21.140, pp. 1–67 (cit. on pp. 2, 4 sq., 12 sq., 34 sqq., 40, 50, 54, 65, 83, 88).

Pranav Rajpurkar, Robin Jia, and Percy Liang (2018). *Know What You Don't Know: Unanswerable Questions for SQuAD*. In: *Proceedings of the 56th Annual Meeting of the Association for*

Computational Linguistics (Volume 2: Short Papers). Melbourne, Australia, pp. 784–789 (cit. on p. 74).

Tharindu Ranasinghe, Constantin Orasan, and Ruslan Mitkov (2019). *Semantic Textual Similarity with Siamese Neural Networks*. In: *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*. Varna, Bulgaria, pp. 1004–1011 (cit. on p. 89).

Sabbir M. Rashid, David De Roure, and Deborah L. McGuinness (2018). *A Music Theory Ontology*. In: *Proceedings of the 1st International Workshop on Semantic Applications for Audio and Music*. Monterey, CA, USA, pp. 6–14 (cit. on p. 2).

Srinivas Ravishankar, June Thai, Ibrahim Abdelaziz, Nandana Mihindukulasooriya, Tahira Naseem, Pavan Kapanipathi, Gaetano Rossillo, and Achille Fokoue (2021). "A Two-Stage Approach towards Generalization in Knowledge Base Question Answering". In: arxiv.org eprint (cit. on pp. 51, 58).

Siva Reddy, Mirella Lapata, and Mark Steedman (2014). "Large-scale Semantic Parsing without Question-Answer Pairs". In: *Transactions of the Association for Computational Linguistics* 2, pp. 377–392 (cit. on p. 4).

Samuel Reyd and Amal Zouaq (2023). *Assessing the Generalization Capabilities of Neural Machine Translation Models for SPARQL Query Generation*. In: *The Semantic Web – ISWC*. Athens, Greece, pp. 484–501 (cit. on p. 92).

S. E. Robertson and S. Walker (1994). *Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval*. In: *SIGIR*. Dublin, Ireland, pp. 232–241 (cit. on p. 54).

Subendhu Rongali, Luca Soldaini, Emilio Monti, and Wael Hamza (2020). *Don't Parse, Generate! A Sequence to Sequence Architecture for Task-Oriented Semantic Parsing*. In: *Proceedings of The Web Conference 2020*. Taipei, Taiwan, pp. 2962–2968 (cit. on p. 44).

Md Rashad Al Hasan Rony, Uttam Kumar, Roman Teucher, Liubov Kovriguina, and Jens Lehmann (2022). "SGPT: A Generative Approach for SPARQL Query Generation From Natural Language Questions". In: *IEEE Access* 10, pp. 70712–70723 (cit. on p. 51).

Amrita Saha, Vardaan Pahuja, Mitesh M. Khapra, Karthik Sankaranarayanan, and Sarath Chandar (2018). *Complex Sequential Question Answering: Towards Learning to Converse Over Linked Question Answer Pairs with a Knowledge Graph*. In: *AAAI*. New Orleans, Louisiana, USA, pp. 705–713 (cit. on p. 52).

Ahmad Sakor, Isaiah Onando Mulang, Kuldeep Singh, Saeedeh Shekarpour, Maria-Esther Vidal, Jens Lehmann, and Sören Auer (2019). *Old is Gold: Linguistic Driven Approach for Entity and Relation Linking of Short Text*. In: *NAACL-HLT (1)*, pp. 2336–2346 (cit. on p. 41).

Gerard Salton and Michael J McGill (1986). "Introduction to Modern Information Retrieval". In: (cit. on p. 14).

Apoorv Saxena, Aditay Tripathi, and Partha Talukdar (2020). *Improving Multi-hop Question Answering over Knowledge Graphs using Knowledge Base Embeddings*. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online, pp. 4498–4507 (cit. on p. 51).

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling (2018). *Modeling Relational Data with Graph Convolutional Networks*. In: *The Semantic Web - ESWC*. Vol. 10843. Heraklion, Greece, pp. 593–607 (cit. on p. 51).

Nathan Schucher, Siva Reddy, and Harm de Vries (2022). *The Power of Prompt Tuning for Low-Resource Semantic Parsing*. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Dublin, Ireland, pp. 148–156 (cit. on pp. 66, 69).

Abigail See, Peter J. Liu, and Christopher D. Manning (2017). *Get To The Point: Summarization with Pointer-Generator Networks*. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada, pp. 1073–1083 (cit. on pp. 12 sq., 23, 40).

Priyanka Sen, Alham Fikri Aji, and Amir Saffari (2022). *Mintaka: A Complex, Natural, and Multilingual Dataset for End-to-End Question Answering*. In: *Proceedings of the 29th International Conference on Computational Linguistics*. Gyeongju, Republic of Korea, pp. 1604–1619 (cit. on p. 74).

Rico Sennrich, Barry Haddow, and Alexandra Birch (2016). *Neural Machine Translation of Rare Words with Subword Units*. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany, pp. 1715–1725 (cit. on p. 29).

Özge Sevgili, Artem Shelmanov, Mikhail Arkhipov, Alexander Panchenko, and Chris Biemann (2022). "Neural Entity Linking: A Survey of Models Based on Deep Learning". In: *Semantic Web* 13, pp. 527–570 (cit. on pp. 2, 86).

Peter Shaw, Ming-Wei Chang, Panupong Pasupat, and Kristina Toutanova (2021). *Compositional Generalization and Natural Language Variation: Can a Semantic Parsing Approach Handle Both?* In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online, pp. 922–938 (cit. on p. 41).

Noam Shazeer and Mitchell Stern (2018). *Adafactor: Adaptive learning rates with sublinear memory cost*. In: *International Conference on Machine Learning*. Stockholm, Sweden, pp. 4596–4604 (cit. on p. 67).

Saeedeh Shekarpour, Edgard Marx, Axel-Cyrille Ngonga Ngomo, and Sören Auer (2015). "SINA: Semantic Interpretation of User Queries for Question Answering on Interlinked Data". In: *Journal of Web Semantics* 30, pp. 39–51 (cit. on pp. 40, 44).

Tao Shen, Xiubo Geng, Tao Qin, Daya Guo, Duyu Tang, Nan Duan, Guodong Long, and Daxin Jiang (2019). *Multi-Task Learning for Conversational Question Answering over a Large-Scale Knowledge Base*. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*. Hong Kong, China, pp. 2442–2451 (cit. on p. 51).

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh (2020). *AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts*. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Online, pp. 4222–4235 (cit. on p. 65).

Yiheng Shu and Zhiwei Yu (2023). "Data Distribution Bottlenecks in Grounding Language Models to Knowledge Bases". In: arxiv.org eprint (cit. on pp. 93 sq.).

Kuldeep Singh, Andreas Both, Dennis Diefenbach, Saedeeh Shekarpour, Didier Cherix, and Christoph Lange (2016). *Qanary – The Fast Track to Creating a Question Answering System with Linked Data Technology*. In: *The Semantic Web*, pp. 183–188 (cit. on p. 40).

Kuldeep Singh, Ioanna Lytra, Arun Sethupat Radhakrishna, Akhilesh Vyas, and Maria-Esther Vidal (2018). *Dynamic Composition of Question Answering Pipelines with FRANKENSTEIN*. In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. Ann Arbor, MI, USA, pp. 1313–1316 (cit. on p. 40).

Kuldeep Singh, Arun Sethupat Radhakrishna, Andreas Both, Saeedeh Shekarpour, Ioanna Lytra, Ricardo Usbeck, Akhilesh Vyas, Akmal Khikmatullaev, Dharmen Punjani, Christoph Lange, Maria Esther Vidal, Jens Lehmann, and Sören Auer (2018). *Why Reinvent the Wheel: Let's Build Question Answering Systems Together*. In: *Proceedings of the 2018 World Wide Web Conference*. Lyon, France, pp. 1247–1256 (cit. on p. 40).

Tommaso Soru, Edgard Marx, Diego Moussallem, Gustavo Publio, André Valdestilhas, Diego Esteves, and Ciro Baron Neto (2017). *SPARQL as a Foreign Language*. In: *Proceedings of the 13th International Conference on Semantic Systems - SEMANTiCS Posters and Demos*. Amsterdam, The Netherlands (cit. on p. 41).

Tommaso Soru, Edgard Marx, André Valdestilhas, Diego Esteves, Diego Moussallem, and Gustavo Publio (2018). *Neural Machine Translation for Query Construction and Composition*.

In: *ICML Workshop on Neural Abstract Machines & Program Induction (NAMPI v2)* (cit. on p. 4).

Elias Stengel-Eskin and Benjamin Van Durme (Sept. 2023). "Calibrated Interpretation: Confidence Estimation in Semantic Parsing". In: *Transactions of the Association for Computational Linguistics* 11, pp. 1213–1231. eprint: https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00598/2161218/tacl_a_00598.pdf (cit. on p. 92).

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit (2019). *Insertion Transformer: Flexible Sequence Generation via Insertion Operations*. In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. New Orleans, LA, USA, pp. 5976–5985 (cit. on p. 5).

Jannik Strötgen and Michael Gertz (2010). *HeidelTime: High Quality Rule-Based Extraction and Normalization of Temporal Expressions*. In: *Proceedings of the 5th International Workshop on Semantic Evaluation*. Uppsala, Sweden, pp. 321–324 (cit. on p. 41).

Yu Su, Huan Sun, Brian Sadler, Mudhakar Srivatsa, Izzeddin Gür, Zenghui Yan, and Xifeng Yan (2016). *On Generating Characteristic-rich Question Sets for QA Evaluation*. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, pp. 562–572 (cit. on p. 75).

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum (2007). *Yago: A Core of Semantic Knowledge*. In: *Proceedings of the 16th International Conference on World Wide Web*. Banff, Alberta, Canada, pp. 697–706 (cit. on p. 1).

Haitian Sun, Tania Bedrax-Weiss, and William Cohen (2019). *PullNet: Open Domain Question Answering with Iterative Retrieval on Knowledge Bases and Text*. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*. Hong Kong, China, pp. 2380–2390 (cit. on p. 51).

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen (2018). *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*. Brussels, Belgium, pp. 4231–4242 (cit. on p. 51).

Tianxiang Sun, Yunfan Shao, Xipeng Qiu, Qipeng Guo, Yaru Hu, Xuanjing Huang, and Zheng Zhang (2020). *CoLAKE: Contextualized Language and Knowledge Embedding*. In: *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online), pp. 3660–3670 (cit. on p. 51).

Weiqi Sun, Haidar Khan, Nicolas Guenon des Mesnards, Melanie Rubino, and Konstantine Arkoudas (2022). *Unfreeze with Care: Space-Efficient Fine-Tuning of Semantic Parsing Models*. In: *Proceedings of the ACM Web Conference 2022*. Virtual Event, Lyon, France, pp. 999–1007 (cit. on p. 66).

Yawei Sun, Lingling Zhang, Gong Cheng, and Yuzhong Qu (2020). "SPARQA: Skeleton-Based Semantic Parsing for Complex Questions over Knowledge Bases". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34, pp. 8952–8959 (cit. on pp. 4, 41).

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le (2014). *Sequence to Sequence Learning with Neural Networks*. In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. Montreal, Canada, pp. 3104–3112 (cit. on p. 21).

Kai Sheng Tai, Richard Socher, and Christopher D. Manning (2015). *Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks*. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China, pp. 1556–1566 (cit. on p. 5).

Alon Talmor and Jonathan Berant (2018). *The Web as a Knowledge-Base for Answering Complex Questions*. In: *Proceedings of the 2018 Conference of the North American Chapter of the*

Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). New Orleans, Louisiana, pp. 641–651 (cit. on p. 75).

Thomas Pellissier Tanon, Marcos Dias de Assunção, Eddy Caron, and Fabian M. Suchanek (2018). *Demoing Platypus - A Multilingual Question Answering Platform for Wikidata*. In: *The Semantic Web: ESWC*. Vol. 11155. Heraklion, Crete, Greece, pp. 111–116 (cit. on p. 51).

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample (2023). *LLaMA: Open and Efficient Foundation Language Models*. arxiv.org eprint (cit. on p. 94).

Dan N. Tran, Laura Pascazio, Jethro Akroyd, Sebastian Mosbach, and Markus Kraft (2023). *Leveraging Text-to-Text Pre-Trained Language Models for Question Answering in Chemistry*. Tech. rep. 313. Preprint Series, Cambridge (cit. on p. 92).

Trieu H. Trinh and Quoc V. Le (2018). "A Simple Method for Commonsense Reasoning". In: arxiv.org eprint (cit. on pp. 35, 43).

Priyansh Trivedi, Gaurav Maheshwari, Mohnish Dubey, and Jens Lehmann (2017). *LC-QuAD: A Corpus for Complex Question Answering over Knowledge Graphs*. In: *The Semantic Web – ISWC 2017*. Vienna, Austria, pp. 210–218 (cit. on pp. 40, 74 sq.).

T. Trouillon, J. Welbl, S. Riedel, and G. Gaussier E.and Bouchard (2016). *Complex Embeddings for Simple Link Prediction*. In: *Proceedings of The 33rd International Conference on Machine Learning*. Vol. 48. New York, New York, USA, pp. 2071–2080 (cit. on p. 89).

Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano (2012). *Template-Based Question Answering over RDF Data*. In: *Proceedings of the 21st International Conference on World Wide Web*. Lyon, France, pp. 639–648 (cit. on p. 4).

Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, Bastian Haarmann, Anastasia Krithara, Michael Röder, and Giulio Napolitano (2017). "7th Open Challenge on Question Answering over Linked Data (QALD-7)". In: *Semantic Web Challenges*. Vol. 769, pp. 59–69 (cit. on p. 75).

Svitlana Vakulenko, Javier David Fernandez Garcia, Axel Polleres, Maarten de Rijke, and Michael Cochez (2019). *Message Passing for Complex Question Answering over Knowledge Graphs*. In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. Beijing, China, pp. 1431–1440 (cit. on p. 51).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). *Attention is All You Need*. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Long Beach, California, USA, pp. 6000–6010 (cit. on pp. 2, 12, 25, 31, 42, 54).

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio (2018). *Graph Attention Networks*. In: *6th International Conference on Learning Representations*. Vancouver, BC, Canada (cit. on p. 52).

Daniel Vollmers, Rricha Jalota, Diego Moussallem, Hardik Topiwala, Axel-Cyrille Ngonga Ngomo, and Ricardo Usbeck (2021). *Knowledge Graph Question Answering Using Graph-Pattern Isomorphism*. In: *Proceedings of the 17th International Conference on Semantic Systems*. Amsterdam, Netherlands, pp. 103–117 (cit. on p. 41).

Denny Vrandečić and Markus Krötzsch (2014). "Wikidata: A Free Collaborative Knowledgebase". In: *Communications of the ACM* 57.10, pp. 78–85 (cit. on pp. 1, 40, 65, 73, 86).

Shuai Wang (2022). *On the Analysis of Large Integrated Knowledge Graphs for Economics, Banking, and Finance*. In: *EDBT/ICDT Workshops*. CEUR Workshop Proceedings, pp. 1–6 (cit. on p. 2).

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang (2021). "KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation". In: *Transactions of the Association for Computational Linguistics* 9, pp. 176–194 (cit. on p. 51).

Yushi Wang, Jonathan Berant, and Percy Liang (2015). *Building a Semantic Parser Overnight*. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China, pp. 1332–1342 (cit. on pp. 66, 75).

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus (2022). "Emergent Abilities of Large Language Models". In: *Transactions of Machine Learning Research*. Survey Certification (cit. on p. 93).

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou (2023). "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models". In: arxiv.org eprint (cit. on p. 94).

Wikidata (2012). *A sample sub-graph from Wikidata KG*. An image. Last accessed on 23.04.2024. URL: https://upload.wikimedia.org/wikipedia/commons/6/60/Linked_Data_-_San_Francisco.svg (cit. on p. 1).

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush (2020). *Transformers: State-of-the-Art Natural Language Processing*. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online, pp. 38–45 (cit. on p. 67).

Yuxiang Wu, Yu Zhao, Baotian Hu, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel (2022). *An Efficient Memory-Augmented Transformer for Knowledge-Intensive NLP Tasks*. In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Abu Dhabi, United Arab Emirates, pp. 5184–5196 (cit. on p. 95).

Wenhan Xiong, Jingfei Du, William Yang Wang, and Veselin Stoyanov (2019). "Pretrained Encyclopedia: Weakly Supervised Knowledge-Pretrained Language Model". In: arxiv.org eprint (cit. on p. 95).

Silei Xu, Shicheng Liu, Theo Culhane, Elizaveta Pertseva, Meng-Hsi Wu, Sina Semnani, and Monica Lam (2023). *Fine-tuned LLMs Know More, Hallucinate Less with Few-Shot Sequence-to-Sequence Semantic Parsing over Wikidata*. In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Singapore, pp. 5778–5791 (cit. on p. 93).

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng (2015). *Embedding Entities and Relations for Learning and Inference in Knowledge Bases*. In: *3rd International Conference on Learning Representations*. San Diego, CA, USA (cit. on pp. 38, 89).

Mohammad Yani and Adila Alfa Krisnadhi (2021). "Challenges, Techniques, and Trends of Simple Knowledge Graph Question Answering: A Survey". In: *Information* 12.7 (cit. on p. 50).

Xi Ye, Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, and Caiming Xiong (2022). *RNG-KBQA: Generation Augmented Iterative Ranking for Knowledge Base Question Answering*. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland, pp. 6032–6043 (cit. on p. 4).

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao (2015). *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 (Volume 1: Long Papers)*. Beijing, China, pp. 1321–1331 (cit. on p. 41).

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh (2016). *The Value of Semantic Parse Labeling for Knowledge Base Question Answering*. In: *Proceedings of*

the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). Berlin, Germany, pp. 201–206 (cit. on p. 75).

Xiaoyu Yin, Dagmar Gromann, and Sebastian Rudolph (2021). "Neural Machine Translating from Natural Language to SPARQL". In: *Future Generation Computer Systems* 117, pp. 510–519 (cit. on p. 5).

Donghan Yu, Chenguang Zhu, Yiming Yang, and Michael Zeng (2022). *JAKET: Joint Pre-training of Knowledge Graph and Language Understanding*. In: *Thirty-Sixth Conference on Artificial Intelligence*. Online, pp. 11630–11638 (cit. on p. 51).

Hamid Zafar, Giulio Napolitano, and Jens Lehmann (2018). *Formal Query Generation for Question Answering over Knowledge Bases*. In: *The Semantic Web - ESWC*. Hersonissos, Greece: Springer International Publishing, pp. 714–728 (cit. on pp. 4 sq., 44).

Luke Zettlemoyer and Michael Collins (2007). *Online Learning of Relaxed CCG Grammars for Parsing to Logical Form*. In: *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Prague, Czech Republic, pp. 678–687 (cit. on p. 4).

Luke S. Zettlemoyer and Michael Collins (2005). *Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars*. In: *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*. Edinburgh, Scotland, pp. 658–666 (cit. on p. 4).

Haoyu Zhang, Jingjing Cai, Jianjun Xu, and Ji Wang (2019). *Complex Question Decomposition for Semantic Parsing*. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy, pp. 4477–4486 (cit. on pp. 4 sq., 38).

Jianan Zhao, Le Zhuo, Yikang Shen, Meng Qu, Kai Liu, Michael Bronstein, Zhaocheng Zhu, and Jian Tang (2023). "GraphText: Graph Reasoning in Text Space". In: arxiv.org eprint (cit. on p. 95).

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi (2023). *Least-to-Most Prompting Enables Complex Reasoning in Large Language Models*. In: *The Eleventh International Conference on Learning Representations, 2023, Kigali, Rwanda* (cit. on p. 94).

Qile Zhu, Haidar Khan, Saleh Soltan, Stephen Rawls, and Wael Hamza (2020). *Don't Parse, Insert: Multilingual Semantic Parsing with Insertion Based Decoding*. In: *Proceedings of the 24th Conference on Computational Natural Language Learning*. Online, pp. 496–506 (cit. on p. 5).

Qile Zhu, Xiyao Ma, and Xiaolin Li (2019). "Statistical learning for semantic parsing: A survey". In: *Big Data Mining and Analytics* 2.4, pp. 217–239 (cit. on p. 4).

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler (2015). *Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books.* In: *Proceedings of the International Conference on Computer Vision*. Santiago, Chile, pp. 19–27 (cit. on pp. 35, 43).

Jianyun Zou, Min Yang, Lichao Zhang, Yechen Xu, Qifan Pan, Fengqing Jiang, Ran Qin, Shushu Wang, Yifan He, Songfang Huang, and Zhou Zhao (2021). *A Chinese Multi-type Complex Questions Answering Dataset over Wikidata*. arxiv.org eprint (cit. on pp. 44, 46).