Conversational Question Answering over Knowledge Graphs with Answer Verbalization

Dissertation
zur
Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität Bonn

von
Endri Kacupaj
aus
Vlore, Albanien

Bonn, 03.2022

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn

1. Gutachter:	Prof. Dr. Jens Lehmann
2. Gutachterin:	Prof. Dr. Maria Maleshkova
Tag der Promotion:	22.11.2022
Erscheinungsjahr:	2022

Abstract

In recent years, publicly available knowledge graphs (KG) have been broadly adopted as a source of knowledge in several tasks such as entity linking, relation extraction, and question answering. Question answering (QA) over knowledge graphs (KG) is an essential task that maps a user's utterance to a query over a KG to retrieve the correct answer. The initial knowledge graph question answering (KGQA) systems were mostly template or rule-based systems with limited learnable modules. Existing research on KGQA has accomplished outstanding results over simple questions, and lately, we have seen a successful effort to improve KGQA for complex questions. However, information needs are not always satisfied in one-shot processing. Either cause the first request is not well-formed, or the user requires more information for a particular topic and issues a series of follow-up questions. In such a conversational scenario, the follow-up questions are often incomplete since they co-reference entities and/or relations from previous interactions. Furthermore, existing KGQA resources and systems do not support answer verbalization. They provide nondescriptive answers extracted from KGs without verbalizing them in natural language utterances.

In this thesis, *Conversational Question Answering over Knowledge Graphs with Answer Verbalization*, we address conversational question answering and answer verbalization tasks while employing knowledge from structured graph data such as knowledge graphs. We present novel approaches based on deep neural network architectures and the multi-task learning paradigm, which allows for improved generalization. First, we propose extending question-answering resources with multiple verbalized answers to study whether the answer verbalization performance can be improved. In this work, we release the first question answering dataset with up to eight paraphrased responses for each question. We provide evaluation baselines to determine our dataset's quality and analyze the performance of various models when trained with one or more paraphrased answers. Next, we explore whether an additional source of information can be incorporated to generate better verbalized answers. Here, we develop a multi-task learning framework that comprises logical forms as auxiliary context alongside the questions. We evaluate on three QA datasets with answer verbalization where results establish a new baseline for the task. Afterward, we continue with the conversational question answering task, where we propose two multi-task neural semantic parsing approaches that construct logical forms and execute them in a knowledge graph to retrieve answers. We present an architecture of stacked pointer networks and we propose employing a Transformer model with Graph Attention Networks for consolidating knowledge graph information. We empirically study the proposed architectural design choices through an extensive evaluation and multiple analyses. Finally, we investigate the impact of incorporating answer verbalization in the conversational question answering task. We present an approach that jointly models the conversational context (entire dialog history with verbalized answers) and knowledge graph paths in a common space for learning joint embedding representations to improve path ranking. Results on standard datasets show a considerable improvement over previous baselines. Our contributions target a broader research agenda by providing efficient, conversational question answering and answer verbalization approaches. All the proposed approaches and resources presented in this thesis are publicly available.

Acknowledgements

This work would not have been possible without the support and guidance of many individuals. First, I would like to thank my supervisor, Prof. Dr. Jens Lehmann, for his constant guidance and support throughout my PhD studies. His insightful feedback pushed me to sharpen my thinking and brought my work to a higher level. Similarly, I would like to express my gratitude and appreciation for Prof. Dr. Maria Maleshkova, whose support and encouragement has been invaluable throughout this thesis. Her motivation, enthusiasm, patience, and insightful guidance helped me see new ideas when tackling the research problem.

I want to extend my sincere thanks to Dr. Kuldeep Singh for co-authoring the majority of research papers and providing support and constructive criticism to improve my research work. I also had great pleasure of working with Joan Plepi, where our collaboration resulted in a core part of this thesis.

I am grateful to the Smart Data Analytics (SDA) group members at the University of Bonn for our great time. I would like to thank my colleagues for their profound and thoughtful discussions: Afshin Sadeghi, Carsten Draschner, Denis Lukovnikov, Dr. Fathoni A. Musyaffa, Firas Kassawat, Dr. Gezim Sejdiu, Dr. Giulio Napolitano, Dr. Günther Kniesel, Dr. Harsh Thakkar, Lars Reimann, Mehdi Ali, Dr. Mohnish Dubey, Mojtaba Nayyeri, and Sebastian Bader. Special thanks to Dr. Hamid Zafar for our wonderful collaboration. I am also grateful to all the ESRs from CLEOPATRA Project for the wonderful time we had during the project meetings. Many thanks to Jason Armitage, Golsa Tahmasebzadeh, Swati, Elisavet Koutsiana, Gullal Singh Cheema, Sara Abdollahi, Sahar Tahmasebi, Gabriel Amaral, Simon Gottschalk, Eric Müller-Budack, and Dr. Sherzod Hakimov, with whom I worked more closely. I also want to thank Shyanmath Premnadh, Barshana Banerjee, Aynur Guluzade, Rishi Tripathi, and Deepansh Pandey, who contributed to my thesis at various stages.

Thanks should also go to Sirma AI, trading as Ontotext, particularly Vladimir Alexiev, Nikola Tulechki, Andrey Tagarev, and Nikola Rusinov, enabling me to visit their offices and observe their daily operations and learn through their projects.

I want to recognize the support in the form of PhD research scholarship that I received from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement no. 812997 (CLEOPATRA), which enabled me to perform my research.

Last but not least, I wish to thank my parents for their support and encouragement throughout my study.

This thesis is dedicated to my parents, Violeta and Agim.

Contents

CHAPTER 1

Introduction

Question answering (QA) interfaces provide a way of querying the information available in various formats, including unstructured (e.g., news articles) and structured (e.g., Knowledge Graphs) data in natural languages. Knowledge Graphs (KGs) have recently garnered significant attention from industry and academia. They are commonly used to curate and connect facts from various information sources available on the web. Existing publicly available large-scale KGs (e.g., DBpedia [1], Wikidata [2], and Yago [3]) have been widely adopted as a reference source of information and knowledge in several fields such as information retrieval and question answering. Knowledge Graph Question Answering (KGQA) systems aim to map a user's natural language question to a query over a KG to retrieve the correct answer. In recent years, a large number of KGQA approaches have been developed [4, 5].

With the increasing popularity of intelligent personal assistants (e.g., Alexa[1], Siri[2], Google Assistant[3], Cortana[4], and others), the research focus has been shifted to conversational question answering (ConvQA) or multi-turn QA. These systems aim to understand the given context and engage users in multi-turn QA to satisfy their information needs. ConvQA over KGs has grasped attention and prominence in recent years owing to the availability of large-scale, multi-turn QA datasets and the advancement of the deep learning field.

While existing KGQA systems provide key technology to facilitate users to query knowledge graphs using natural language, they do not consider interpreting the answer in a more readily comprehended format. In other words, these systems deliver answers as extracted from the knowledge graphs without supporting any answer interpretability method such as verbalizing them into natural language sentences. A recent work [6] has examined different answer interpretation strategies for KGQA systems, such as query representation, graph representation, controlled language representation, and answer verbalization representation. They concluded that verbalizing KGQA answers into natural language is more convenient for the users to understand the answer and its correlation with the question, especially for complex queries. Hence, answer verbalization is an essential element that existing KGQA systems lack and, therefore, a research gap we address in this thesis.

This thesis explores the fields of conversational question answering over knowledge graphs and answer verbalization to introduce novel solutions for advancing the state-of-the-art.

[1] https://developer.amazon.com/en-US/alexa
[2] https://www.apple.com/siri/
[3] https://assistant.google.com/
[4] https://www.microsoft.com/en-us/cortana

Like single-turn QA, multi-turn QA or ConvQA systems comprise multiple components that handle different sub-tasks. Together, these sub-tasks are coordinated to achieve the decisive task of retrieving answers for a given natural language question. The most common sub-tasks include Natural Language Understanding (NLU), Named Entity Recognition (NER), Entity Linking (EL), Relation Linking (RL), and Query Builder (QB). Typically, the NLU module is employed to understand the structure of the input and exploit the semantics of the question. The NER module identifies all entities in the input question, while the EL module links them with KG entities. Besides, the RL module links the relations mentioned in the question with the KG. Finally, the QB module utilizes all this information to identify the most suitable KG path for answering the question. Most approaches employ the QB module to generate a formal query representation of the question and execute it in the underlying KG for extracting the answers. In the end, the retrieved answer is presented to the user. Nowadays, with the emerging of the deep learning field, all these sub-tasks are built using various deep neural network architectures and trained through the availability of large-scale corpora.

Several works [6–10] target the KGQA task by implementing stand-alone components. However, in this way, there is no information shared between the components, and each one is performed individually. While this can generally achieve adequate performance, being more focused on single tasks, we risk ignoring information that might help us improve the overall QA performance. To avoid this, we can share information (e.g., representations) between related tasks by jointly training them. In this way, we enable the broader QA architecture to generalize better on the overall task. This approach is called Multi-Task Learning (MTL) and has emerged in multiple fields in recent years [11]. MTL is our primary learning paradigm for all the proposed models in this thesis.

1.1 Motivation

In recent years, there has been a trend shift from single-turn QA to conversational QA. Most of the existing KGQA approaches refer to single-turn QA, where the users ask a single question and receive answers extracted from the underlying KG. However, in a real-world scenario, users tend to ask complex questions that require multi-turn interaction with the KG. Furthermore, users also pose questions that refer to information from previous interactions with the system. These cases refer to the ConvQA task, where we have to incorporate and analyze the conversational history in order to answer follow-up questions. Figure 1.1 illustrates an example of ConvQA over KGs.

Some common scenarios of follow-up questions are the “incomplete” or “indirect” cases since they require co-reference or ellipsis resolution to handle them. Considering the examples in Figure 1.1, we see the first question “Where was the President of the United States born?”. This is a direct question similar to the single-turn QA. The system here is expected to answer “Scranton, Pennsylvania”. On the second turn, the user asks, “Where did he graduate from?”. As we can see, the question does not contain any entity itself, but it refers to the entity from the previous question (Q1). Interestingly the entity in the first turn is not directly mentioned in the question but instead referred to as the “President of the United States”. For this example, the entity is “Joe Biden”. Therefore, to answer the second question, the system requires performing co-reference resolution employing the conversational context available (question, answer, and entities) from the previous turn. We can see that using only the question is insufficient since we require the entities. Next, in the third turn, we see a similar type of question. The question here, “What year was it established?” refers to the answer entity from the previous turn (“Syracuse University”). Again co-reference resolution is required here for accurately

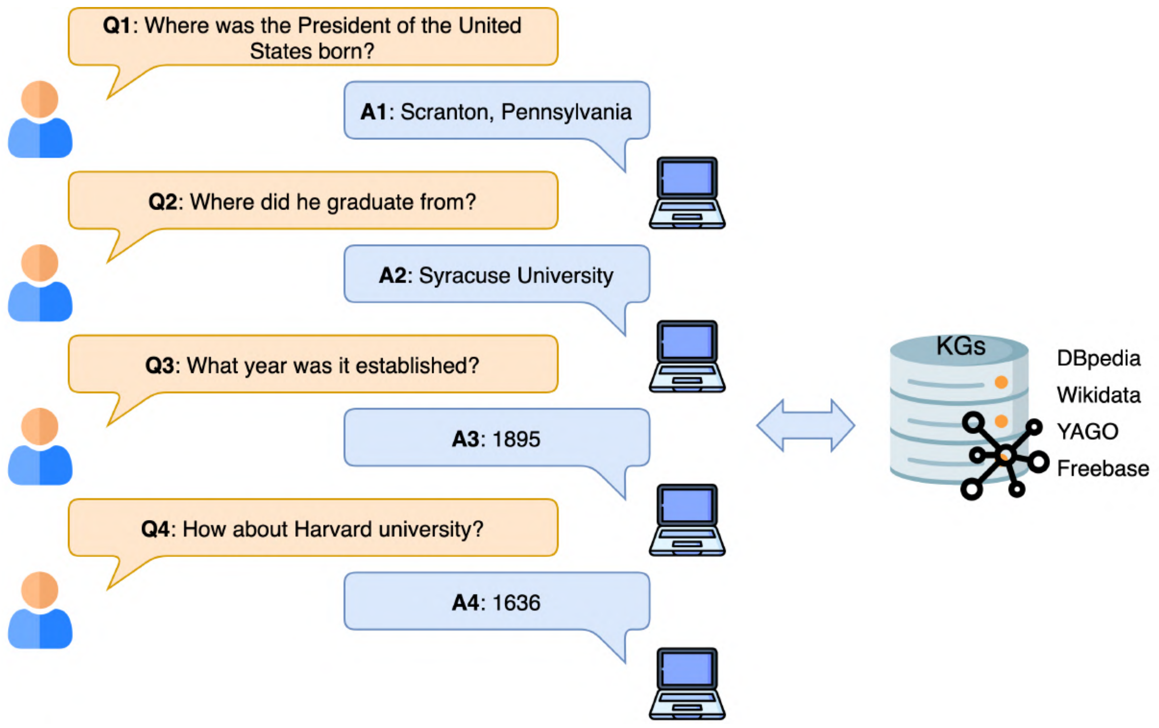
Q1: Where was the President of the United States born?
A1: Scranton, Pennsylvania
Q2: Where did he graduate from?
A2: Syracuse University
Q3: What year was it established?
A3: 1895
Q4: How about Harvard university?
A4: 1636
KGs
DBpedia
Wikidata
YAGO
Freebase

Figure 1.1: A conversational question answering example with answers derived from knowledge graphs.

answering the question. Finally, in the last turn, we can see a different type of question. The user asks "How about Harvard university?". As we can observe, the question contains an entity ("Harvard university"); however, no other information is given. The intent of this question is extracted from the previous interaction (Q3), and to answer such types of questions, we need to perform ellipsis resolution. For single-turn QA systems, the user would have to ask the complete question such as "What year was Harvard University established?" ignoring previous interactions with the system.

In ConvQA, incomplete and indirect questions contain various difficulties. Above, we described two of them that require co-reference and ellipsis resolution. However, there can exist more complicated scenarios where the system must ask so-called "clarification" questions for collecting more information. With clarification questions, we can clarify part of the question with the user in order to provide an answer. The system often chooses to ask a clarification question after a co-reference resolution step is not successful or clear. We can envision a scenario where the user's question refers to entities from the previous turn where multiple entities of the same type exist. Here, the system must clarify for which entity the user desires an answer. For example, if for the question "Where did he graduate from?" the system would answer with both alma maters that Joe Biden graduated (e.g., University of Delaware and Syracuse University). Then for the following question, "What year was it established?" the system would first have to clarify whether the user wants an answer for the University of Delaware or Syracuse University or even both.

Similar to single-turn KGQA, ConvQA over KGs can also be performed via semantic parsing [12–15] or information retrieval-based approaches [16, 17]. The main difference between single-turn KGQA and ConvQA is how we identify all the relevant context required to answer a question. For the ConvQA task, we can obtain the relevant context by incorporating the existing conversational history.

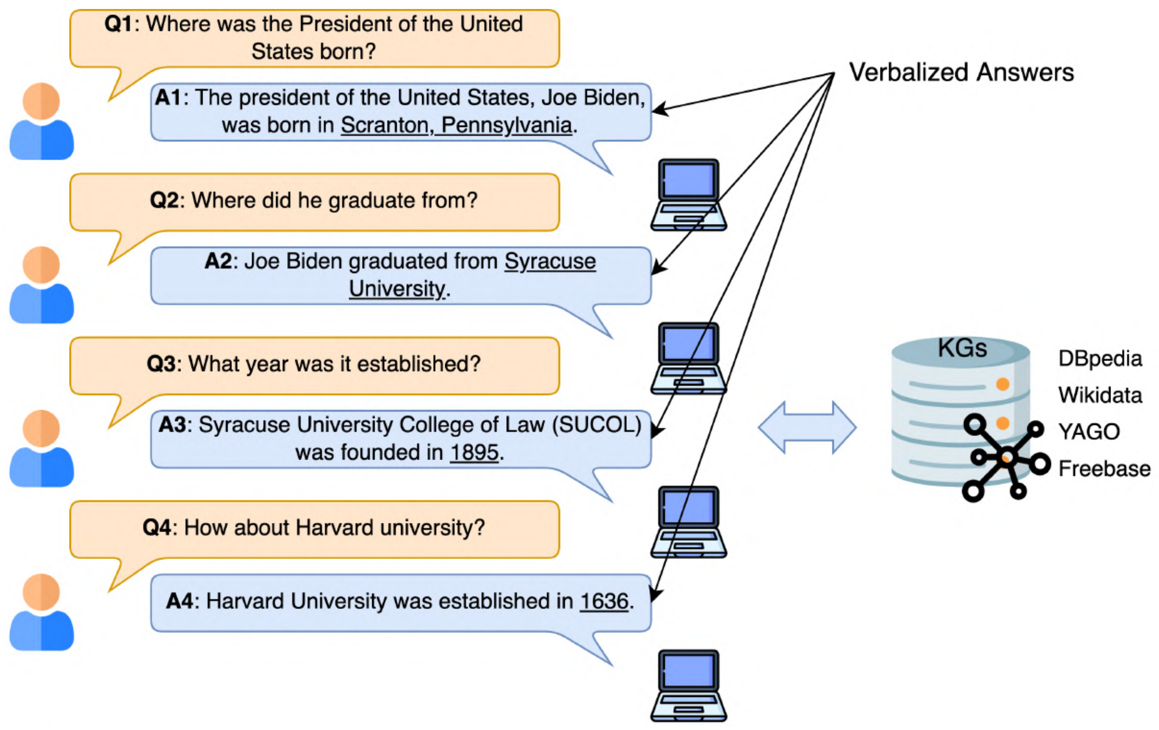
Q1: Where was the President of the United States born?
A1: The president of the United States, Joe Biden, was born in Scranton, Pennsylvania.
Q2: Where did he graduate from?
A2: Joe Biden graduated from Syracuse University.
Q3: What year was it established?
A3: Syracuse University College of Law (SUCOL) was founded in 1895.
Q4: How about Harvard university?
A4: Harvard University was established in 1636.
Verbalized Answers
KGs
DBpedia
Wikidata
YAGO
Freebase

Figure 1.2: A conversational question answering example with verbalized answers.

The work in this thesis mainly focuses on the ConvQA over KGs task, and therefore the mentioned scenarios are addressed in various approaches we propose [14, 15, 18].

Existing KGQA (including ConvQA) systems are only capable of retrieving answers for given questions. However, these answer responses might leave the end-user unsatisfied with their representation. For instance, assuming that the answer from the question "Where was the President of the United States born?" (cf. Figure 1.1) is not known by the user. When the QA system only provides a name with no further explanation, the user might refer to an external source to verify the answer. In an attempt to allow the users to verify the response provided by a QA system, researchers employ various techniques such as (i) graphically visualizing the formal query [19], (ii) exposing the generated formal query [20], and (iii) verbalizing the formal query [21–23]. However, a more sophisticated approach to allow users to validate the answers given by the QA system would be to verbalize the answer to convey the information requested by the user and include additional characteristics that indicate how it was determined. For instance, the answer verbalization for the example question can be "The president of the United States, Joe Biden, was born in Scranton, Pennsylvania." and given this verbalization, the user can better verify that the system is retrieving a location that is the birthplace of "Joe Biden" who is the "President of the United States". At the same time, verbalizing the answers could also be helpful for the ConvQA system's performance. For example, in Figure 1.2, we can see the same conversation as in Figure 1.1 extended with verbalized answers. We can observe that for the question in the second turn (Q2) "Where did he graduate from?", the system can more naturally perform co-reference resolution and identify that the pronoun "he" refers to "Joe Biden" since the verbalized answer reveals him as the "President of the United States".

To this end, we distinguish three advantages that verbalized answers provide compared to coarse answers extracted from KGs: i) express and present better and more natural responses to users, engaging them in conversation. ii) Allow users to validate the answers since they provide additional textual context relevant to the question. iii) They can be helpful for ConvQA systems by supporting them in particular conversational scenarios where the co-reference resolution is not straightforward (cf. Figure 1.2).

Therefore, the work in this thesis addresses the answer verbalization task, which can be correlated with building better ConvQA systems.

1.2 Problem Statement and Challenges

In this section, we define the problem definition for this thesis work, and we look into various challenges that we have to address.

Research Problem Definition

How can we employ multi-task learning to improve the performance of conversational question answering over knowledge graphs and answer verbalization?

We identify several fundamental challenges to be tackled while working towards our research problem. We divide the challenges in three main categories: *Answer Verbalization, ConvQA* and *MTL Challenges*.

1.2.1 Answer Verbalization Challenges

Challenge 1: Lack of KGQA resources that support answer verbalization

Currently, an insufficient number of KGQA datasets support answer verbalization. Furthermore, existing datasets [24, 25] are restricted to only one verbalized answer. However, an answer can be formulated differently using various paraphrases and still containing the same semantic meaning. Having more paraphrased answers can introduce more flexibility and intuitiveness in the conversations.

Challenge 2: Additional context for improving answer verbalization performance

The answer verbalization task aims to generate fluent, natural language answers to a user questions. Currently, only the question is utilized as a source of information to generate verbalized responses. However, a question can have different reformulations that might affect the learning process of the underlying model, especially if no designated patterns are identified for generating the verbalized answers. Therefore, an additional context is required to allow the model to determine that questions with the same meaning posed differently can be satisfied with the same or similar responses.

1.2.2 ConvQA Challenges

Challenge 3: Semantic parsing grammar for ConvQA

Semantic parsing is one of the main methodologies for approaching KGQA. For semantic parsing, we require a grammar of actions to build executable queries over the KG. These actions can vary based on the types of questions and their complexity. While for KGQA, various grammars have been proposed for semantic parsing, they cannot be directly applied to the ConvQA task since they do not address conversational scenarios such as clarification questions. Hence, it is challenging to build a grammar with efficient actions that can be employed for ConvQA.

Challenge 4: Comprise KG information in ConvQA

We perform ConvQA by retrieving answers from KGs. Therefore, incorporating KG information such as entities, relations, and types is crucial for our task. We need to discover methods that determine the relevant information in the KG and incorporate it effectively in our model. Since KG facts are connected, the challenge will also be to exploit correlations for more satisfactory performance.

Challenge 5: Incorporate all available conversational contexts

As stated in our motivation, conversational context plays a vital role in improving ConvQA performance. In particular, the entire dialog history is required to answer conversational questions. Moreover, verbalized answers can provide context that might be helpful for particular scenarios (cf. Figure 1.2). Therefore, we want to recognize all the available conversational contexts (e.g., dialog history, answer verbalization, and possibly more) that are valuable and effectively incorporate them.

Challenge 6: Consolidate answer verbalization into overall ConvQA architecture

Yet, another challenge is to integrate the answer verbalization task with the ConvQA task. Our goal is to merge answer verbalization sub-tasks with ConvQA sub-tasks under a unified framework. However, the outputs of the tasks are different and identifying solutions for MTL training challenges, like the two we describe below, is not clear. We must determine how to establish a connection between the sub-tasks for effectively sharing training signals and controlling the learning process of each sub-task for optimal performance on both primary tasks (Answer Verbalization and ConvQA).

1.2.3 Multi-Task Learning Challenges

Challenge 7: Identify information to share among sub-tasks

The ConvQA task consists of multiple sub-tasks (e.g., NLU, NER, QB) that need to be coordinated for answering questions from KGs. For MTL, the goal is to share training signals of related sub-tasks to improve generalization for the overall ConvQA task. Here, the challenge is identifying what information (i.e., representations) to share between the sub-tasks for improving the QA task's overall performance.

Challenge 8: Coordinate training of multiple diverse sub-tasks

Another challenge related to MTL is the coordination of training multiple QA sub-tasks. The complexity of each sub-task has to be identified and, accordingly, control the model's learning process. For instance, we can imagine that the model will require multiple iterations over the training data for more complicated sub-tasks to achieve the best performance. While for less complicated sub-tasks, fewer iterations might be sufficient. In such cases, we endanger overfitting training data in less complicated sub-tasks. Here, we desire to recognize the complexity of each task and depict this in the learning process for achieving a more promising overall generalization for the QA task.

1.3 Research Questions

In order to investigate the challenges mentioned above, we break down the main research question into four more concrete and specific questions. Figure 1.3 illustrates the connections between the four research questions and the main research question.

Research Question 1 (*RQ1*)

How do multiple paraphrased answers affect the performance of answer verbalization?

As a first step, we want to address the lack of KGQA resources that support answer verbalization. This would allow us to focus on building better answer verbalization systems. While constructing the resource, we want to provide paraphrases of generated verbalizations to introduce more flexibility and see whether the models can leverage this for improving their performance. To develop such a resource, we require building a semi-automated framework with numerous steps that assures the grammatical correctness of the generated natural language answers. By working with *RQ1*, we are addressing the first challenge described above.

Research Question 2 (*RQ2*)

How can we incorporate logical forms to improve the performance of answer verbalization via multi-task learning?

Here, we aim to advance the state-of-the-art for verbalizing answers of KGQA systems. In order to do that, we want to build a multi-task learning framework for answer verbalization. As mentioned in the second challenge, we must further identify an additional context to support the verbalization task. Our goal is to employ logical forms as a supplementary context of the question and develop sub-tasks that can be jointly trained to produce the final verbalized answer. Therefore, we must also assess the MTL challenges for developing the framework.

Research Question 3 (*RQ3*)

How can we develop better and more efficient multi-task semantic parsing approaches for conversational question answering?

Semantic parsing is a key methodology for several state-of-the-art QA systems. Here, we want to perform semantic parsing for ConvQA via MTL. For this, we need to develop multi-task frameworks and semantic grammars from which we generate logical forms and execute them in a KG for retrieving the answers. With this research question, we address the ConvQA challenges regarding the semantic parsing grammar and comprising KG information in the task. At the same time, both MTL challenges are addressed.

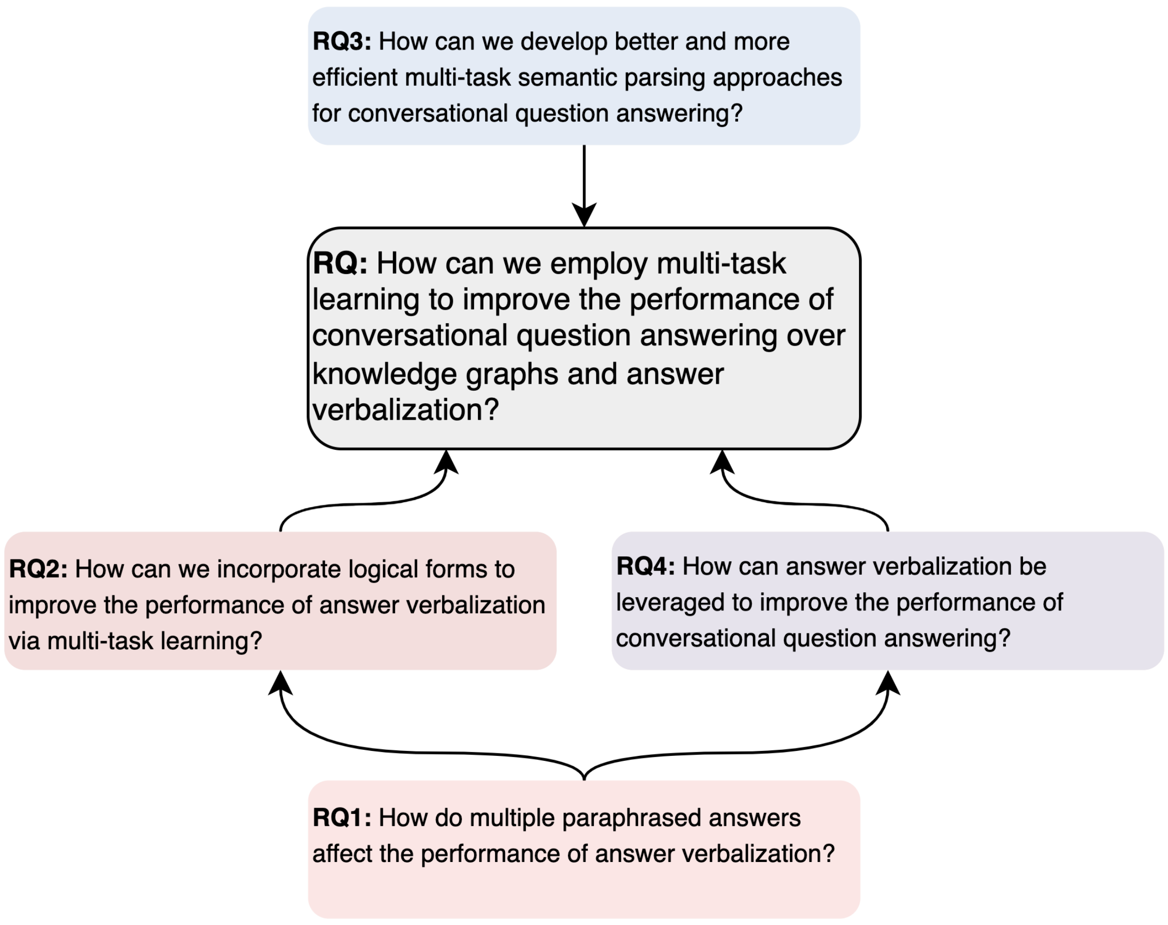
RQ3: How can we develop better and more efficient multi-task semantic parsing approaches for conversational question answering?
RQ: How can we employ multi-task learning to improve the performance of conversational question answering over knowledge graphs and answer verbalization?
RQ2: How can we incorporate logical forms to improve the performance of answer verbalization via multi-task learning?
RQ4: How can answer verbalization be leveraged to improve the performance of conversational question answering?
RQ1: How do multiple paraphrased answers affect the performance of answer verbalization?

Figure 1.3: Demonstrating the connection of the four research questions to the main research question.

Research Question 4 (*RQ4*)

How can answer verbalization be leveraged to improve the performance of conversational question answering?

Finally, we want to investigate the possibility of including the answer verbalization task into the ConvQA architecture. In this way, we can leverage answer verbalization information as part of the conversational context for improving ConvQA performance. This research question is inspired by the motivational example we described (cf. Figure 1.2), where answer verbalization presents more natural responses to users and can also be helpful for seeking answers for the ConvQA task where additional context might be required. In order to answer *RQ4*, we trigger several challenges mentioned above, such as the fifth and sixth challenges regarding the available conversational context and consolidating the answer verbalization task in ConvQA. Similar to the two previous research questions, both MTL challenges are addressed here.

1.4 Thesis Overview

This section highlights the main contributions to the dissertation and the research areas investigated. We refer to the scientific publications published during the research period covering those contributions.

1.4.1 Contributions

Our contributions cover a spectrum of research areas in the scope of answer verbalization and conversational question answering over knowledge graphs via multi-task learning paradigm, as depicted in Figure 1.4.

Contributions for *RQ1*

A question answering resource extended with multiple paraphrased answers and empirical analysis to measure the effectiveness.

Corresponding work: Kacupaj et.al [26]

To address the research question *RQ1*, we first develop a KGQA resource that supports answer verbalization. We provide a semi-automated framework for generating multiple paraphrase responses for a question by utilizing methods such as back-translation. Furthermore, we release the first KGQA dataset with multiple paraphrased verbalized responses. In particular, the dataset consists of up to eight unique paraphrased answers for each question and employs DBpedia [1] as the underlying KG. Moreover, we provide evaluation baselines that serve to confine our dataset's quality and define a benchmark for future research. Finally, we examine the performance of various models when trained with one or more paraphrased responses to determine the impact of multiple verbalized answers on the answer verbalization performance.

Contributions for *RQ2*

A multi-task learning framework for verbalizing answers using questions and logical forms as inputs.

Corresponding work: Kacupaj et.al [27]

For the second research question, *RQ2*, we present the first multi-task-based answer verbalization framework that employs questions and logical forms as inputs and simultaneously trains four modules for generating natural language answers. Furthermore, in this framework, we propose a novel similarity threshold and cross attention modules to determine the relevance between the inputs and fuse information. We validate the framework capabilities by providing an extensive evaluation and ablation study on three QA datasets with answer verbalization, including the generated resource for *RQ1*. Evaluation results establish a new baseline for the answer verbalization task that will drive future research in a newly studied problem.

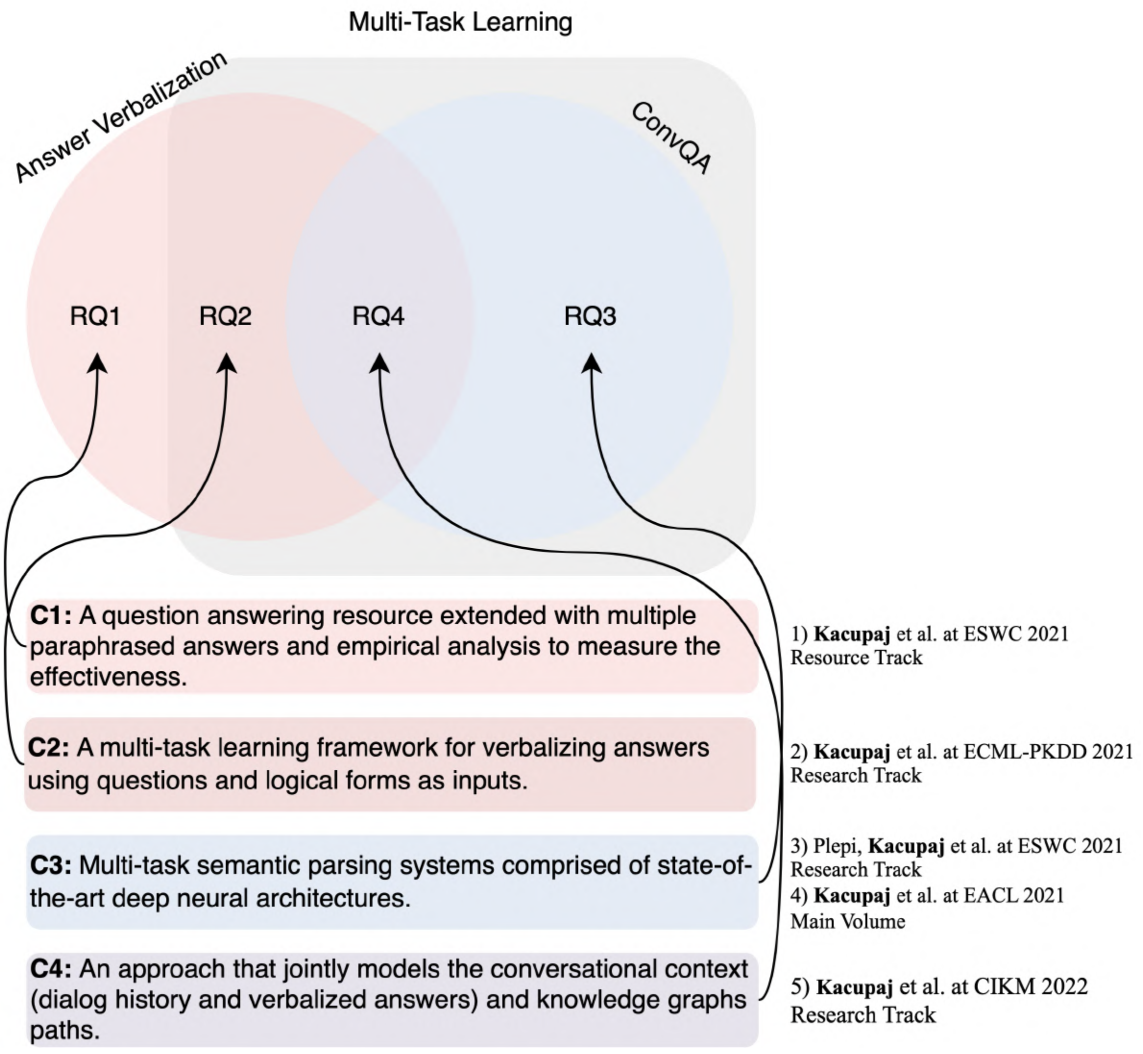
Multi-Task Learning
Answer Verbalization
ConvQA
RQ1
RQ2
RQ4
RQ3
C1: A question answering resource extended with multiple paraphrased answers and empirical analysis to measure the effectiveness.
C2: A multi-task learning framework for verbalizing answers using questions and logical forms as inputs.
C3: Multi-task semantic parsing systems comprised of state-of-the-art deep neural architectures.
C4: An approach that jointly models the conversational context (dialog history and verbalized answers) and knowledge graphs paths.
1) **Kacupaj** et al. at ESWC 2021 Resource Track
2) **Kacupaj** et al. at ECML-PKDD 2021 Research Track
3) Plepi, **Kacupaj** et al. at ESWC 2021 Research Track
4) **Kacupaj** et al. at EACL 2021 Main Volume
5) **Kacupaj** et al. at CIKM 2022 Research Track

Figure 1.4: Illustrating the contributions to the research questions and the scope they address. Next to contributions, we indicate associated works.

Contributions for *RQ3*

Multi-task semantic parsing systems comprised of state-of-the-art deep neural architectures.

Corresponding works: Plepi, Kacupaj et.al [14] & Kacupaj et.al [15]

For the third research question, *RQ3*, we develop two multi-task neural semantic parsing approaches for (complex) conversational question answering over knowledge graphs. We distinguish the works from the number of sub-tasks they perform and the different deep neural architectural modules they utilize. Furthermore, in the first work, we present a reusable grammar for neural semantic parsing that defines various actions that can be combined to develop logical forms and execute them on a KG to fetch answers. We also propose an architecture of stacked pointer networks for incorporating knowledge graph information on the task. In the second work, we modify certain grammar actions for optimal performance, and we employ a Transformer model supplemented with Graph Attention Networks to exploit the correlations between (entity) types and predicates due to its message-passing ability between the nodes. We further propose a novel entity recognition module that detects, links, filters, and permutes all relevant entities. We empirically study the proposed architectural design choices through extensive evaluations, ablation studies, and multiple analyses. Overall, both approaches have achieved state-of-the-art results on several question types from a large-scale conversational question answering dataset.

Contributions for *RQ4*

An approach that jointly models the conversational context (dialog history and verbalized answers) and knowledge graphs paths.

Corresponding work: Kacupaj et.al [18]

For the last research question, *RQ4*, we study whether the availability of entire dialog history, domain information, and verbalized answers can act as context sources in determining the ranking of KG paths while retrieving correct answers. We propose an approach that models conversational context and KG paths in a shared space by jointly learning the embeddings for homogeneous representation. For evaluation, we extend standard ConvQA datasets to support answer verbalization using a similar approach with the contribution in *RQ1*. We systematically study the impact of incorporating additional context on the performance. Results indicate that our method outperforms existing baselines on all domains and overall, where in particular cases, the margin of ranking metrics is even more significant compared to the state-of-the-art performance. Our evaluation results establish a new baseline, which we believe will drive future research in a new way of developing such frameworks.

The contributions we present in this thesis identify solutions for all the challenges mentioned and supply answers to our main research question. Overall, two of the contributions directly address the answer verbalization task, while the other two the conversational question answering task. We employ multi-task learning in three of the contributions as our primary learning paradigm. Finally, each contribution significantly advances the state-of-the-art in the respective field.

1.4.2 Publications

The following publications contribute a scientific basis to this thesis and serve as a reference point for numerous figures, tables, and ideas presented in the later chapters.

- ***Conference Papers (peer reviewed)***
    1. **Endri Kacupaj**, Joan Plepi, Kuldeep Singh, Harsh Thakkar, Jens Lehmann, and Maria Maleshkova. "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, pp. 850-862. 2021. DOI: 10.18653/v1/2021.eacl-main.72

2. **Endri Kacupaj**, Shyamnath Premnadh, Kuldeep Singh, Jens Lehmann, and Maria Maleshkova. “VOGUE: Answer Verbalization Through Multi-Task Learning.” In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 563-579. Springer, Cham, 2021. DOI: 10.1007/978-3-030-86523-8_34
3. Joan Plepi, **Endri Kacupaj**, Kuldeep Singh, Harsh Thakkar, and Jens Lehmann. “Context Transformer with Stacked Pointer Networks for Conversational Question Answering over Knowledge Graphs.” In European Semantic Web Conference, pp. 356-371. Springer, Cham, 2021. DOI: 10.1007/978-3-030-77385-4_21
4. **Endri Kacupaj**, Barshana Banerjee, Kuldeep Singh, and Jens Lehmann. “ParaQA: A Question Answering Dataset with Paraphrase Responses for Single-Turn Conversation.” In European Semantic Web Conference, pp. 598-613. Springer, Cham, 2021. DOI: 10.1007/978-3-030-77385-4_36
5. **Endri Kacupaj**, Hamid Zafar, Jens Lehmann, and Maria Maleshkova. “VQuAnDa: Verbalization question answering dataset.” In European Semantic Web Conference, pp. 531-547. Springer, Cham, 2020. DOI: 10.1007/978-3-030-49461-2_31
6. **Endri Kacupaj**, Kuldeep Singh, Maria Maleshkova, and Jens Lehmann. “Contrastive Representation Learning for Conversational Question Answering over Knowledge Graphs.” In Proceedings of the 31st ACM International Conference on Information & Knowledge Management, pp. 925-934. 2022. DOI: 10.1145/3511808.3557267
7. Jason Armitage, **Endri Kacupaj**, Golsa Tahmasebzadeh, Maria Maleshkova, Ralph Ewerth, and Jens Lehmann. “MLM: A benchmark dataset for multitask learning with multiple languages and modalities.” In Proceedings of the 29th ACM International Conference on Information & Knowledge Management, pp. 2967-2974. 2020. DOI: 10.1145/3340531.3412783

- ***Demo Papers (peer reviewed)***

8. Golsa Tahmasebzadeh, **Endri Kacupaj**, Eric Müller-Budack, Sherzod Hakimov, Jens Lehmann, and Ralph Ewerth. 2021. “GeoWINE: Geolocation based Wiki, Image, News and Event Retrieval.” In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR ’21). Association for Computing Machinery, New York, NY, USA, 2565–2569. DOI: 10.1145/3404835.3462786

- ***Workshop Articles (peer reviewed)***

9. Simon Gottschalk, **Endri Kacupaj**, Sara Abdollahi, Diego Alves, Gabriel Amaral, Elisavet Koutsiana, Tin Kuculo et al. “OEKG: The Open Event Knowledge Graph.” In CLEOPATRA@ WWW, pp. 61-75. 2021. DOI: Vol-2829/paper5

- ***Miscellaneous Papers (peer reviewed)***

  Following publication originated during the thesis but is not part of the thesis itself.

10. Aynur Guluzade, **Endri Kacupaj**, and Maria Maleshkova. “Demographic Aware Probabilistic Medical Knowledge Graph Embeddings of Electronic Medical Records.” In International Conference on Artificial Intelligence in Medicine, pp. 408-417. Springer, Cham, 2021. DOI: 10.1007/978-3-030-77211-6_48

The list of publications completed during the PhD term is available in Appendix A.

1.5 Thesis Outline

The structure of the thesis consists of the following seven chapters:

Chapter 1 – Introduction: Introduces the thesis describing the motivation, main research problem, challenges, and contributions that address the four research questions. It further lists the scientific publications that constitute the scientific basis of this dissertation.

Chapter 2 – Background: Presents the background covering the broad fields of Knowledge Graphs and Question Answering. It further describes multiple deep neural architectures and the primary learning paradigm (Multi-Task Learning) employed in this thesis to align the context and provide a holistic overview of the research problem.

Chapter 3 – Extending Question Answering Resources to Support Answer Verbalization: Addresses the first research question (***RQ1***) concerned with question answering resources supporting answer verbalization by proposing a framework that extends an existing question answering dataset to support answer verbalization via multiple verbalized answers. It also provides a new resource named "*ParaQA*" and illustrates experiments with various deep learning models.

Chapter 4 – Answer Verbalization through Multi-Task Learning: Addresses the second research question (***RQ2***) concerned with improving answer verbalization by proposing the "*VOGUE*" framework, which attempts to generate a verbalized answer using a hybrid approach (questions and queries as inputs concurrently) through a multi-task learning paradigm. It presents results on existing datasets for answer verbalization, where "*VOGUE*" outperforms all baselines on both BLEU and METEOR metrics.

Chapter 5 – Conversational Question Answering through Multi-Task Learning: Addresses the third research question (***RQ3***) concerned with developing improved and efficient multi-task semantic parsers for conversational question answering over knowledge graphs by presenting two frameworks, "*CARTON*" and "*LASAGNE*," which consists of multiple modules that parse the input conversation and generate grammar-based logical forms for retrieving the answer from the underlying knowledge graph.

Chapter 6 – Conversational Question Answering with Answer Verbalization: Addresses the fourth and last research question (***RQ4***) concerned with improving path ranking for conversational question answering by proposing "*PRALINE*", an approach that jointly models the conversational context (entire dialog history, domain information, and verbalized answers) and KG paths to learn homogeneous embedding representations for improving the path ranking. It further provides experiments on a ConvQA dataset extended with answer verbalization, on which "*PRALINE*" outperforms baselines on all domains and overall.

Chapter 7 – Conclusion and Future Directions: Concludes the work of this dissertation by summarizing the core contributions and their impact on the community and lays out the direction of future work.

CHAPTER 2

Background

This chapter provides the principles and concepts that lay the foundations for addressing the research problem defined in Chapter 1. We will first look into the Knowledge Graph as a well-structured source of information. Following, we continue with Question Answering over Knowledge Graph, where we describe the two most common methods. Next, we will look at Deep Neural Network Architectures that we employ across different approaches in this thesis. In the end, we describe the Multi-Task Learning paradigm and present existing approaches and mechanisms.

2.1 Knowledge Graph

A knowledge graph can be described as a structured representation of entities, relations, and semantic descriptions. Entities are real-world objects and abstract concepts, and relations describe the relation between entities. Semantic descriptions of entities and their relations contain types and properties with a well-defined meaning. Google introduced the term knowledge graph in 2012 [28]. According to [29], the term knowledge graph is synonymous with a knowledge base with a minor difference. A knowledge graph is a graph when considering its graph structure [30]. While, when it includes formal semantics, it can be considered a knowledge base for interpretation and inference over facts [31]. The knowledge is expressed in the form of a triple (subject, predicate, object) or (head, relation, tail) under the resource description framework (RDF) [32], which is also interpreted as a directed graph with entities as nodes and relations as edges. As of a definition, Paulheim [33] stated:

Definition 2.1.1 (Knowledge Graph)**.** A knowledge graph, 1) mainly describes real-world entities and their interrelations, organized in a structured graph; 2) defines possible classes and relations of entities in a schema; 3) allows for potentially interrelating arbitrary entities with each other and 4) covers various topical domains.

Another work [34], stated that a knowledge graph acquires and integrates information into an ontology and employs a reasoner to derive new knowledge. A work from Wang et al. [35] proposed a knowledge graph as a multi-relational graph composed of entities and relations that are perceived as nodes and different types of edges, respectively. Today knowledge graphs are used in various domains and applications [36–38].

2.2 Question Answering over Knowledge Graph

Large-scale KGs have been constructed to serve many downstream tasks. Applications for KGs include semantic search [39–43], recommendations [44–46], chatbots [47–50], information extraction [51–55], etc. [36–38, 56]. Based on available KGs, Knowledge Graph Question Answering (KGQA) is a task that aims to answer natural language questions with KGs as its knowledge source. Early approaches to KGQA [57–61] focus on answering simple questions involving only a single fact. For example, the simple question "*Where was Albert Einstein born?*" can be answered using only the fact "*(Albert Einstein, BornIn, German Empire)*". Recently, researchers have started paying more attention to answering complex questions over KGs [62, 63]. Complex questions involve multiple entities, expressing compound relations and including numerical operations. There exist two mainstream methods for performing question answering. First, both methods identify the topic entity in a question and link it to an entity in the KG. Then they obtain the answers within the neighborhood of the topic entity by either executing a parsed logical form or reasoning in a question-specific graph extracted from the KG (the "*sub*"-graph is usually extracted with the topic entity). The two approaches are known as semantic parsing and information retrieval. Figure-2.1 provides an abstract illustration of how they function. We further describe how the two approaches operate in the following subsections.

2.2.1 Semantic Parsing

The semantic parsing-based approaches represent a question by a symbolic logical form and then execute it against the knowledge graph to obtain the final answers. Overall, these methods aim at parsing a natural language utterance into a logical form [65–67]. They usually operate in the following steps:

1. A natural language understanding (NLU) module is employed to comprehend parts of the question by conducting a semantic and syntactic analysis and obtaining the encoded representation.
2. A parsing module is used to transform the encoded representation into an uninstantiated logical form. The logical form is usually only a syntactic representation of the question without any KG information (e.g., entities, relations). A pre-defined grammar with several actions is required to generate the logical forms, which vary based on the requirements.
3. The logical form is instantiated and validated by conducting semantic alignments via KG grounding.
4. The final generated logical form is executed against the KG to retrieve the predicted answer.

Semantic parsing-based methods provide an interpretable reasoning process since they generate expressive logical forms. The performance heavily relies on the vocabulary design and the selected modules for the parsing algorithm.

2.2.2 Information Retrieval

The information retrieval-based approaches construct a question-specific graph that contains comprehensive information related to the question and ranks all the paths in the extracted graph based on their relevance. Fundamentally, the information retrieval-based approaches directly retrieve and rank answers from the KG, considering the information conveyed in the question [57, 58]. They operate in the following steps:

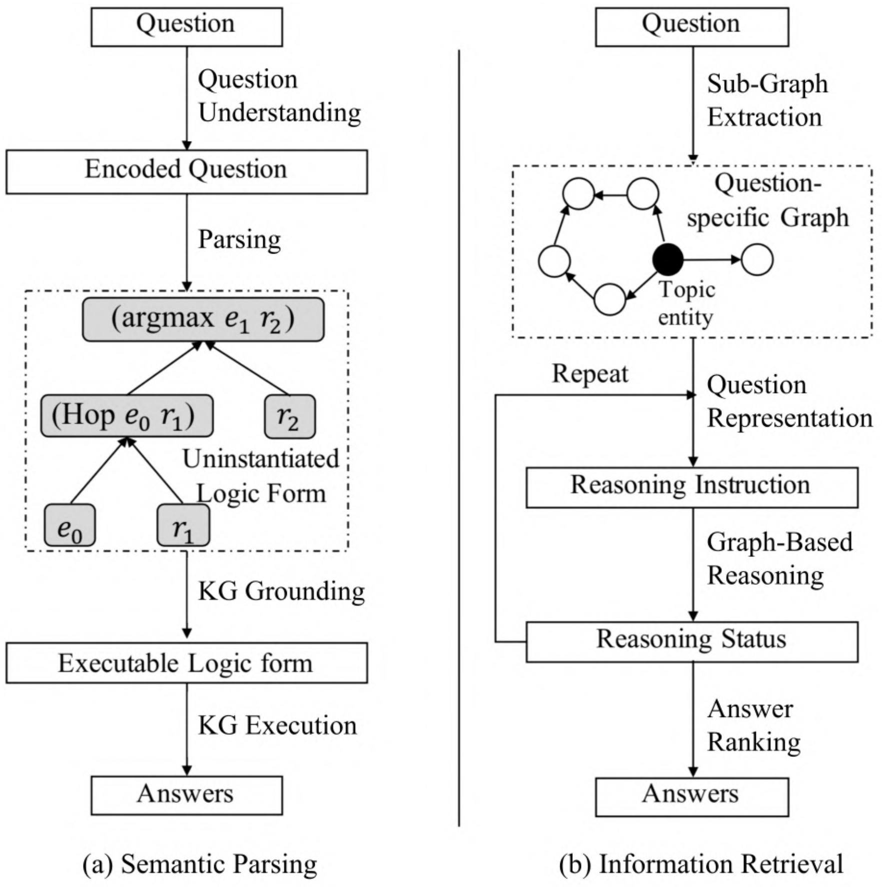

Figure 2.1: An abstract illustration of how semantic parsing and information retrieval based KGQA systems operate (Source [64]).

1. Given the topic entity, extract a question-specific graph from the KG. Usually, this graph includes all question-related entities and relations as nodes and edges, respectively.
2. An encoder module encodes the input question.
3. A graph-based module conducts semantic matching through a vector-based computation to propagate and aggregate the neighboring entities' information within the graph. The reasoning status, which can vary (e.g., entity distribution, relation representations), is updated based on reasoning instructions. Steps two and three are also performed multiple times to improve reasoning [68, 69].
4. Finally, a ranking module is employed to rank the entities in the graph based on the reasoning status. The top-ranked entities are considered as the predicted answers to the question.

Information retrieval-based approaches perform complex reasoning on graph structure and semantic matching. Such a paradigm naturally fits into end-to-end training and makes these methods easier to train. However, the style of reasoning here is less interpretable compared to semantic parsing-based methods.

2.3 Deep Neural Network Architectures

Deep learning is a subset of a larger family of machine learning techniques based on representation learning and artificial neural networks. Deep learning architectures like deep neural networks, deep reinforcement learning, recurrent neural networks, and convolutional neural networks have been used in domains like computer vision, speech recognition, and natural language processing. Natural Language Processing (NLP) is one of the popular branches of artificial intelligence that allows machines to understand, manipulate or respond to a human in their natural language. With the hype of deep learning, most NLP tasks have been performed through various neural architectures. In particular, Question Answering (QA) is a discipline within the field of NLP. In this thesis, we build conversational QA systems by employing or building such neural architectures. In this section, we describe various neural architectures we used.

2.3.1 Sequence to Sequence Networks

Sequence to sequence (seq2seq) is a method of encoder-decoder-based language processing approaches that maps an input sequence to an output sequence with independent length. Seq2seq models are currently used for several NLP tasks, such as machine translation [70, 71], question answering [72, 73], text summarization [74, 75], speech recognition [76, 77], etc. Here we describe the seq2seq model from [78].

The most common seq2seq models employ a Recurrent Neural Network (RNN) to encode the input (source) sentence into a single vector, also referred a context vector. The context vector is an abstract representation of the entire input sentence. Next, a second RNN receives as input the context vector and learns to produce the output (target) sentence by generating it one word at a time.

The objective of the encoder-decoder model is to estimate the conditional probability $p(y_1, ..., y_{T'}|x_1, ..., x_T)$ where $X = \{x_1, x_2, ..., x_T\}$ is an input sequence and $Y = \{y_1, y_2, ..., y_{T'}\}$ is the corresponding target sequence whose length T' may differ from T. The seq2seq model computes this conditional probability by first obtaining the fixed dimensional representation z of the input sequence $\{x_1, x_2, ..., x_T\}$ given by the last hidden state of the encoder, and then computing the probability of $\{y_1, y_2, ..., y_{T'}\}$ with a standard decoder whose initial hidden state is set to the representation z of $\{x_1, x_2, ..., x_T\}$:

$$p(y_1, ..., y_{T'}|x_1, ..., x_T) = \prod_{t=1}^{T'} p(y_t|z, y_1, ..., y_{t-1}).$$

Figure 2.2 shows how seq2seq models operate. Here, the input sequence, "*a b c*", is first passed through an embedding layer and then into the encoder. Helper tokens indicating the beginning (e.g. <*SOS*>) or end (e.g. <*EOS*>) of the sequence are frequently used. For each time-step, the encoder's input is the embedding e, of the current word, $e(x_t)$, as well as the hidden state from the previous time-step/word, h_{t-1}. At the end of the sequence, the encoder outputs a new hidden state h_t. This hidden state can be used as a vector representation of the sentence so far. We can represent the encoder as a function of $e(x_t)$ and h_{t-1}:

$$h_t = \text{encoder}(e(x_t), h_{t-1}).$$

The input sequence can be denoted as $X = \{x_1, x_2, ..., x_T\}$, where x_1 = a, x_2 = b, etc. The initial hidden state, h_0, is usually initialized to zeros. After the final word of the sequence is encoded, the

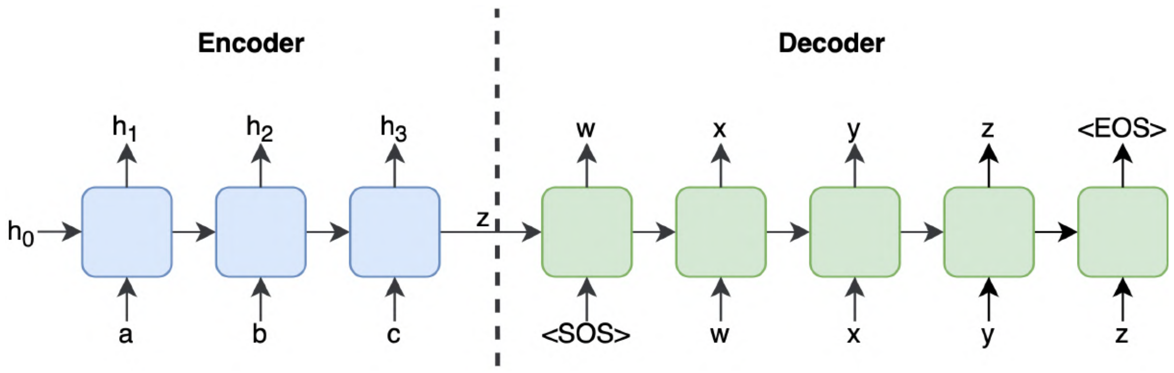

Figure 2.2: A sequence to sequence (seq2seq)/encoder-decoder model example. It reads an input sentence "*a b c*" and produces "*w x y z*" as the output sentence.

final hidden state h_T is used as the context vector, i.e. $h_T = z$ and hence is the vector representation of the entire input sequence.

With the context vector z, the decoding process is initiated in order to get the target sequence ("*w x y z*"). For each time-step, the decoder's input is the embedding d of the current word $d(y_t)$ alongside the hidden state from the previous time-step s_{t-1}. The initial hidden state s_0 is the context vector, $s_0 = z = h_T$, from the encoder. Therefore, similar to the encoder, we define the decoder as:

$$s_t = \text{decoder}(d(y_t), s_{t-1}).$$

The two embedding layers used e, d for the encoder and the decoder, respectively, are different layers without sharing parameters.

As the last step, for predicting a word at each time-step, a linear layer f is trained to receive an input s_t and predict $\hat{y}_t$ as the next word. This can be denoted as:

$$\hat{y}_t = f(s_t).$$

The decoder always generates words one after another, where for each word, the decoder uses $y_{t>1}$ previous predictions. Here, a technique called "*teacher forcing*" is often used, wherewith a probability p the decoder input can be the ground truth next word (y_t) in the sequence. Otherwise, the word predicted by the decoder, $\hat{y}_{t-1}$ is used. At this stage, the predicted target sequence $\hat{Y} = \{\hat{y}_1, \hat{y}_2, ..., \hat{y}_{T'}\}$, is compared against the actual target sentence, $Y = \{y_1, y_2, ..., y_{T'}\}$, to calculate our loss and proceed with training of the model.

The seq2seq process mentioned above can be operated with any recurrent architecture, such as an RNN, an LSTM (Long Short-Term Memory) [79], or a GRU (Gated Recurrent Unit) [80].

2.3.2 Pointer Networks

Pointer networks operate as a seq2seq model by addressing the case where the sequential output data contain discrete elements that depend on the variable input size, specifically when the length of the input data is undetermined. Overall, a pointer network learns the conditional probability of a target sequence with discrete element tokens corresponding to positions in the source sequence (cf. Figure 2.3). The issue of variable vocabulary size is tackled by employing additive attention [70]. While the

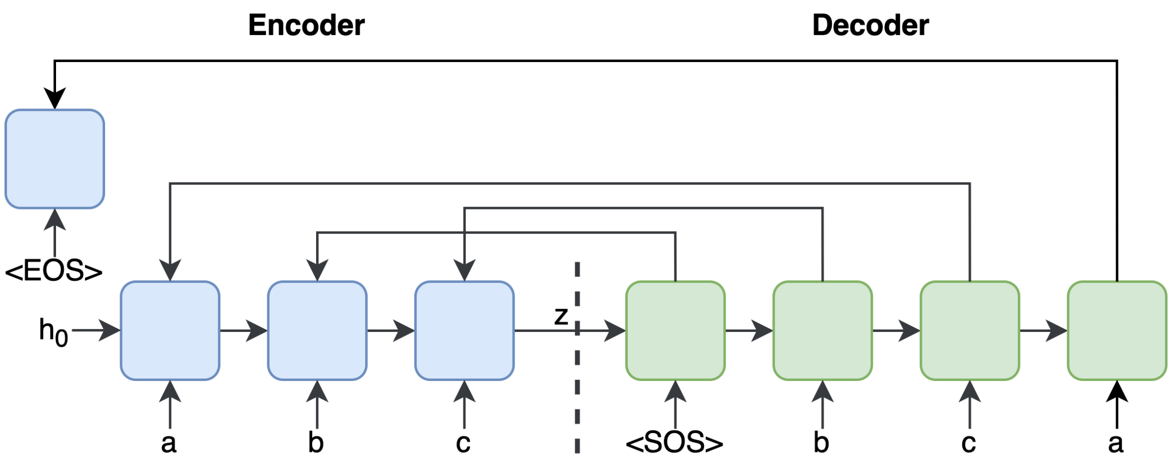

Figure 2.3: A pointer network model example. It reads an input sentence "*a b c*" and each decoder step selects an input element as the output sequence.

authors [70] propose attention to combine hidden states of the encoder to a context vector for each decoder step, pointer networks utilize the same attention as a pointer to choose an element from the input sequence.

More precisely, a simple adjustment of the additive attention model allows applying the method in cases where the output vocabulary depends on the elements in the input sequence. Typically, a seq2seq model employs a softmax over the fixed-sized vocabulary to compute $p(y_1, ..., y_{T'}|x_1, ..., x_T)$. Therefore it cannot handle a variable vocabulary size. For pointer networks, to compute $p(y_1, ..., y_{T'}|x_1, ..., x_T)$, the additive attention is modified as:

$$\begin{aligned} \omega_j^i &= v^\top tahn(\boldsymbol{W}_1 h_j + \boldsymbol{W}_2 s_i) \\ p(y_1, ..., y_{T'}|x_1, ..., x_T) &= softmax(\omega_j^i) \end{aligned} \tag{2.1}$$

where v, $\boldsymbol{W}_1$, and $\boldsymbol{W}_2$ are learnable parameters, h_j and s_i are the encoder and decoder hidden states, respectively. The softmax normalizes ω_j^i to produce the output distribution over the vocabulary of the input sequence. Here, the encoder hidden state h_j is not combined to propagate further information to the decoder; it rather uses ω_j^i as pointers to the input tokens.

While original pointer networks where proposed to only select elements from the input sequence, the attention modification they perform is adopted in different tasks where the vocabulary size is not pre-defined.

2.3.3 Transformer Networks

A Transformer is an encoder-decoder (seq2seq) neural architecture that avoids recurrence and relies entirely on an attention mechanism to represent dependencies between input and output. Before, the seq2seq models were based on recurrent [79, 80] or convolutional [81] neural networks extended with attention mechanisms [70, 71]. The Transformer employs only attention mechanisms, allowing significantly more parallelization compared to RNNs and CNNs.

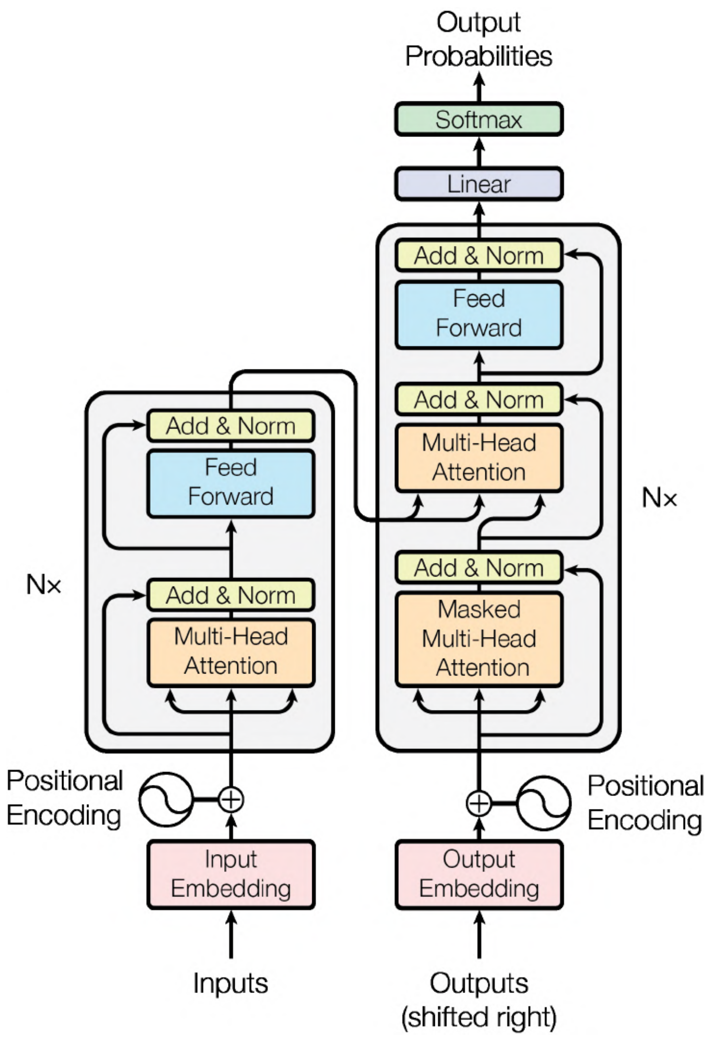

Figure 2.4: Transformer architecture (Source [82]).

Figure 2.4 illustrates the transformer architecture. In the Transformer, the encoder consists of multiple blocks that process the input sequence iteratively one after another. The decoder consists of blocks that do the same to the encoder's output. Each encoder block aims to generate representations that describe which input tokens are relevant to each other. In comparison, the decoder block receives all the encoder representations and the embedded output (shifted to the right by one position) to generate the output sequence. For achieving this, each encoder and decoder block employs attention mechanisms. For each input token, the attention mechanism weighs the relevance and correlation with every other input token to produce the output. A decoder block contains an additional attention mechanism layer used to process the outputs information from the previous block and before receiving the representations from the encoder. Both encoder and decoder blocks contain feed-forward layers for additional processing and include residual connections and normalization layers.

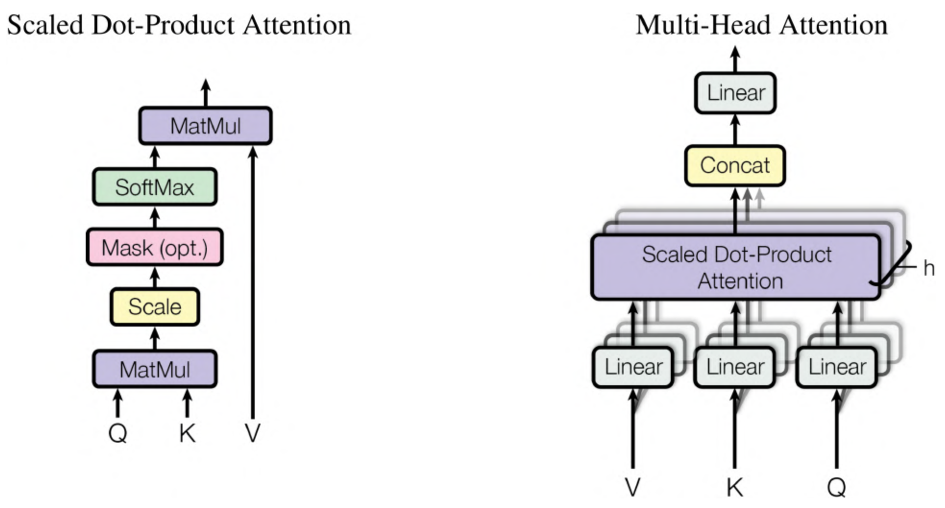

Figure 2.5: Scaled Dot-Product Attention and Multi-Head Attention utilized in Transformer (Source [82]).

Scaled Dot-Product Attention

Each Transformer block contains a so called "*Scaled Dot-Product Attention*" (cf. Figure 2.5). The inputs in this attention mechanism consist of a set of queries Q and keys K of dimension d_k, and a set of values V of dimension d_v. First are computed the dot products of the query Q with all keys K and divided by $\sqrt{d_k}$, after a softmax function is applied to obtain the weights on the values V. Formally, the matrix output is computed as:

$$Attention(Q, K, V) = softmax(\frac{QK^\top}{\sqrt{d_k}})V. \tag{2.2}$$

Here, considering that $q \in \mathbb{R}^{d_k}$ and $k \in \mathbb{R}^{d_k}$ are vectors of dimension d_k with components as independent random variables with mean 0 and variance 1, then the dot product, $q \cdot k = \sum_{i=1}^{d_k} u_i v_i$, has mean 0 and variance d_k. Since a variance of 1 is preferred, they are divided by d_k.

Multi-head Attention

This module that runs through a *Scaled Dot-Product Attention* several times in parallel (cf. Figure 2.5). The attention outputs are concatenated and linearly transformed into the desired dimension. Intuitively, multiple attention heads allow focusing on different parts of the sequence (e.g., shorter-term dependencies vs. longer-term dependencies). The multi-head attention is defined as:

$$\begin{gathered} MultiHead(Q, K, V) = [head_1, \ldots, head_h]\boldsymbol{W}^O, \\ \text{where } head_i = Attention(Q\boldsymbol{W}_i^Q, K\boldsymbol{W}_i^K, V\boldsymbol{W}_i^V), \end{gathered} \tag{2.3}$$

where $\boldsymbol{W}_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $\boldsymbol{W}_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $\boldsymbol{W}_i^V \in \mathbb{R}^{d_{model} \times d_v}$, and $\boldsymbol{W}^O \in \mathbb{R}^{hd_v \times d_{model}}$ and are learnable matrices. d_{model} is the dimension of the model.

Applications

Nowadays, the Transformer model is considered a breakthrough in the NLP field. It has been gradually applied to tasks such as machine translation [83] and time series prediction [84]. Other NLP tasks that use Transformers include summarization [85], question answering [86], named entity recognition [87], and many more. It is already utilized in various tasks with different input and output modalities, such as text↔image [88], text↔audio [89], video understanding [90], and biological sequence analysis [91]. Besides, in this thesis, we mainly employ a Transformer model as a base of our approaches.

Language Models

An essential component of modern natural language processing is a language model. Language modeling is a statistical method for predicting words based on human language patterns. Language models are used in NLP-based applications for several tasks, including spell correction, speech recognition, question answering, sentiment analysis, summarization, etc.

The Transformer architecture we described above is commonly used for building state-of-the-art language models such as BERT [92], BART [93], and GPT [94]. These models employ the encoder part of the Transformer (e.g., BERT) or the decoder part (e.g., GPT), or even both (e.g., BART), and are pre-trained on a huge amount of corpora for various tasks. After pre-training, the models are fine-tuned for any specific task and dataset.

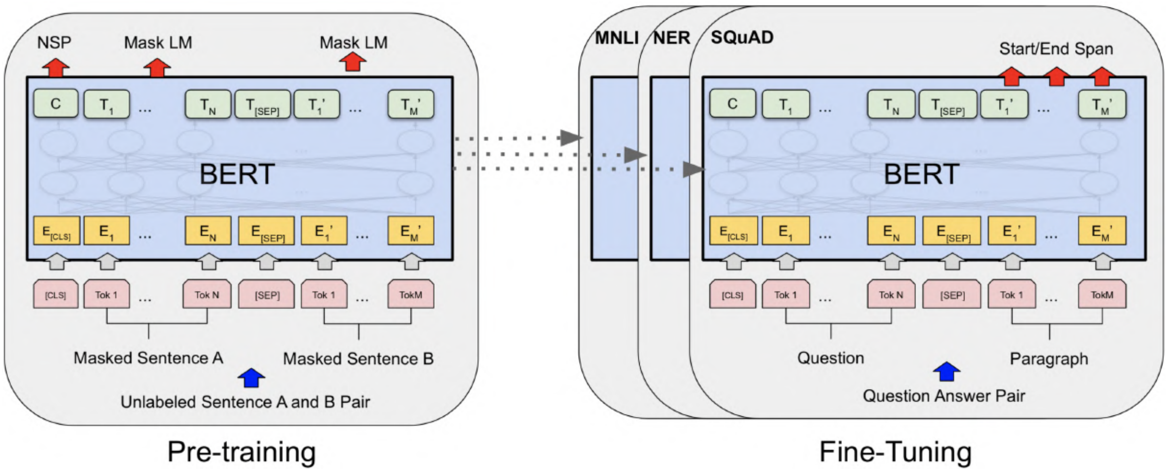

Figure 2.6: Pre-training and fine-tuning procedures for BERT (Source [92]).

BERT. The BERT (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers) [92] model consists of Transformers bidirectional encoder, and it was pre-trained with BookCorpus [95] and English Wikipedia[1] (2,5 billion words) on two tasks: i) a language modeling task where 15% of the input tokens were masked, and BERT had to predict them from the surrounding context, ii) a next sentence prediction binary task, where BERT was trained to predict if a chosen B sentence is the actual one that comes after an A sentence. BERT is fine-tuned for different tasks such as Natural Language Understanding, Reading Comprehension Question Answering, and Multiple Choice, showing new state-of-the-art results. Figure 2.6 illustrates the pre-training and fine-tuning process of BERT model.

[1] https://en.wikipedia.org/

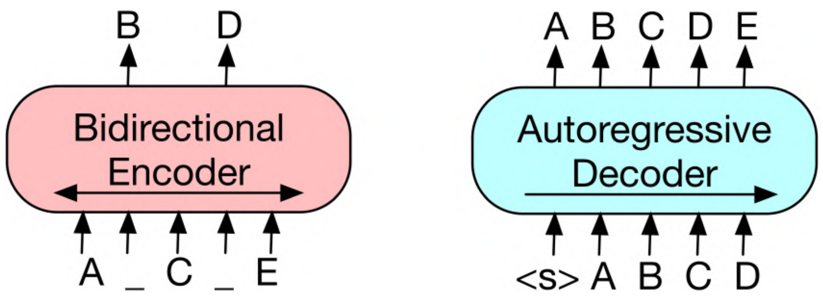

Figure 2.7: (left) BERT bidirectional encoder. (right) GPT autoregressive decoder (Source [93]).

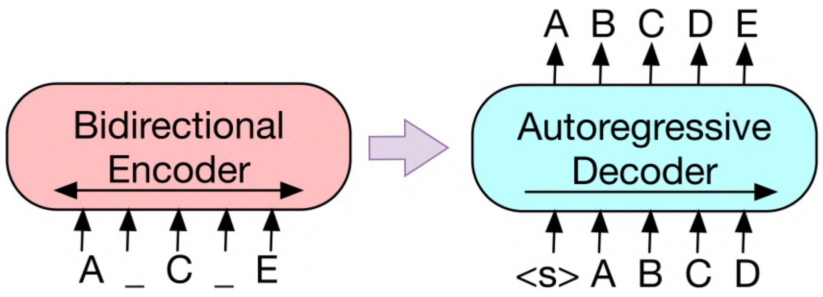

Figure 2.8: BART architecture with BERT bidirectional encoder and GPT autoregressive decoder (Source [93]).

GPT. Another model, GPT (**G**enerative **P**re-**T**raining) [94], consists of a Transformer autoregressive decoder allowing it to handle also natural language generation tasks. While BERT encodes the input bidirectionally, GPT can only condition on leftward context, so it can not learn bidirectional interactions (cf. Figure 2.7). Like BERT, GPT operates in two steps: i) pre-train on a standard language modeling objective, ii) supervised fine-tuning on a dataset and task-specific objective.

BART. Another state-of-the-art Transformer-based model is BART [93], a denoising autoencoder for pre-training seq2seq models. BART is pre-trained by corrupting the input sequence with an arbitrary noising function, and it learns to reconstruct the original text. It consists of a BERT bidirectional encoder with a fully visible attention mask and a GPT left-to-right decoder with a casual attention mask (cf. Figure 2.8).

With the success of BERT, pre-trained Transformer-based language models became a standard method for approaching and solving a majority of NLP tasks. This thesis uses the BERT model in different approaches for extracting initial representations [14, 15, 18]. We further employ BART as the base module of a broader architecture for conversational QA with answer verbalization [18].

2.3.4 Graph Attention Networks

Graphs can model sets of objects, also referred to as nodes in the graph, and their relationships referred to as edges between the nodes. Research on graph data tasks with machine learning has gained increased popularity in the last few years due to the expressive power of graphs. They have been used in different fields such as natural science for physical systems [96, 97], protein-protein interaction networks [98], social science for social networks [99], citation networks [100], knowledge graphs [101], and many others [102]. As a non-Euclidean data structure for machine learning,

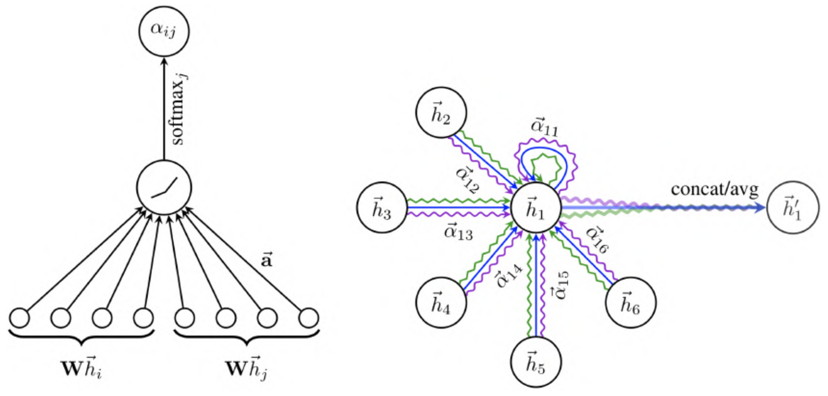

Figure 2.9: (left) The attention mechanism employed by GAT. (right) Multi-head attention in GAT (Source [107]).

analyzing graphs includes node classification, link prediction, and clustering tasks. Recently, deep learning-based approaches employ Graph Neural Networks (GNNs) [103] that operate on the graph domain to handle those tasks. Several GNN architectures have been proposed to work with graph data and employ different learning modules such as convolutional neural networks [104–106] and attention networks [107, 108]. From the attention-based approaches, a model called Graph Attention Network (GAT) [107] aims to generalize the attention mechanism [70, 82] on graphs and incorporate it into the propagation step. A GAT module (cf. Figure 2.9) stacks layers in which nodes can attend over their neighborhoods' features. It implicitly allows defining different weights to different nodes in a neighborhood without requiring costly matrix operation (e.g., inversion) or depending on knowing the graph structure upfront.

Formally, a GAT computes the hidden states of each node by attending to its neighbors via a self-attention strategy. For a node v, the hidden state is obtained by:

$$\mathbf{h}_v^{t+1} = \rho\left(\sum_{u\in\mathcal{N}_v} \alpha_{vu}\boldsymbol{W}\mathbf{h}_u^t\right), \quad \alpha_{vu} = \frac{exp(LeakyReLU(\mathbf{a}^T[\boldsymbol{W}\mathbf{h}_v \parallel \boldsymbol{W}\mathbf{h}_u]))}{\sum_{k\in\mathcal{N}_v} exp(LeakyReLU(\mathbf{a}^T[\boldsymbol{W}\mathbf{h}_v \parallel \boldsymbol{W}\mathbf{h}_k]))}, \tag{2.4}$$

where ρ is an alternative non-linear function, $\mathcal{N}_v$ is the neighborhood set of node v, $\mathbf{h}_u^t$ is the hidden state of neighborhood node u at time step t, $\boldsymbol{W}$ is the weight matrix associated with the linear transformation applied to each node, and a is the weight vector of a multi-layer perceptron (MLP).

Furthermore, the GAT model employs multi-head attention [82] to stabilize the learning process. It utilizes K independent attention head matrices to calculate the hidden states and then concatenates their features or computes the average, resulting in the representations:

$$\text{Concatenation: } \mathbf{h}_v^{t+1} = \parallel_{k=1}^{K}\sigma\left(\sum_{u\in\mathcal{N}_v} \alpha_{vu}^k \boldsymbol{W}_k\mathbf{h}_u^t\right), \quad \text{Average: } \mathbf{h}_v^{t+1} = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{u\in\mathcal{N}_v} \alpha_{vu}^k \boldsymbol{W}_k\mathbf{h}_u^t\right). \tag{2.5}$$

Here α_{vu}^k is the normalized attention coefficient computed by the k-th attention head. $\parallel$ is the vector concatenation operation, and σ the logistic sigmoid function. Overall, the attention architecture has several characteristics: i) the calculation of the neighbor node pairs is parallelizable and therefore, the operation is efficient; ii) it can be employed to graph nodes with different degrees by specifying arbitrary weights to neighbors; iii) it is also suitable for inductive learning problems.

In this thesis, we employ a GAT model for ConvQA [15], where we aim to identify several KG elements (i.e., entity types and relations) relevant to the input conversation.

2.4 Multi-Task Learning

With machine learning, we generally train models to perform single tasks. However, for complex tasks that consist of several steps or sub-tasks, using models that focus only on one task, we endanger ignoring essential and relevant information that might help us with overall performance. Alternatively, sharing information between the related tasks enables the model to generalize better on the primary task. This approach is also known as "*Multi-Task Learning*" (MTL). Formally we define multi-task learning as:

Definition 2.4.1 (Multi-Task Learning)**.** Given m learning tasks $\{T_i\}_{i=1}^m$ where all the tasks or a subset are related, multi-task learning aims to learn the m tasks simultaneously to improve the performance of a model for each task T_i by utilizing the training signals contained in all or some of the tasks.

According to Caruana [109], "Multi-task learning is an inductive transfer mechanism whose principal goal is to improve generalization performance.". Essentially, multi-task learning employs information contained in the training signals of related tasks to enhance generalization. This is achieved via parallelly training several tasks while using shared representations. Furthermore, the training signals between the tasks can serve as an inductive bias. Overall, we perform multi-task learning as soon as we aim to *optimize more than one loss function* and whether or not we use shared representations.

2.4.1 Approaches

Nowadays, multi-task learning is also referred to as joint learning and has been successfully employed in different fields and areas [110–113]. In the deep learning era, multi-task learning is commonly performed in two ways.

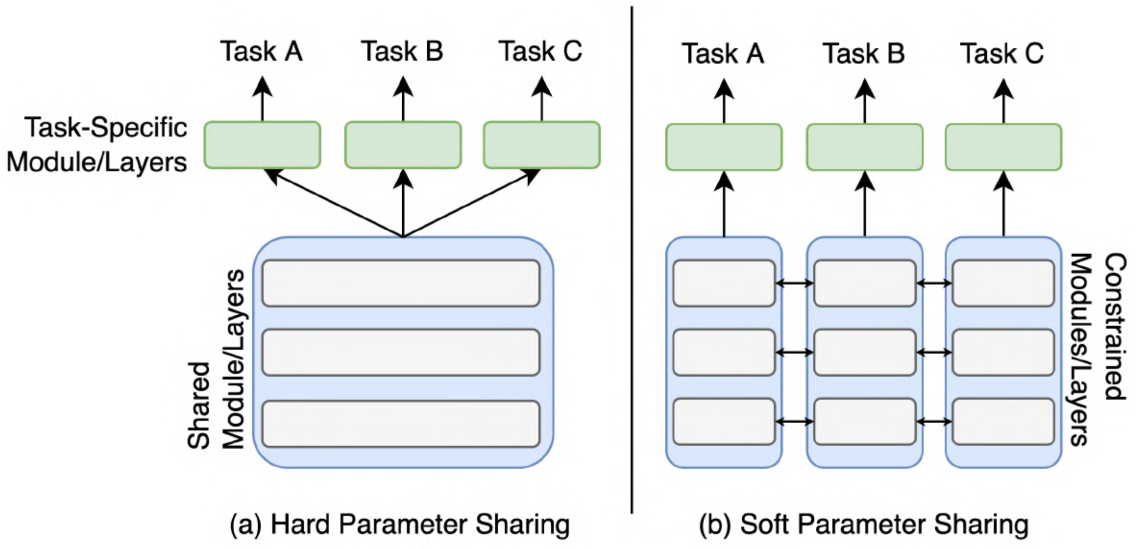
Task A
Task B
Task C
Task-Specific Module/Layers
Shared Module/Layers
(a) Hard Parameter Sharing
Task A
Task B
Task C
Constrained Modules/Layers
(b) Soft Parameter Sharing

Figure 2.10: Parameter sharing techniques for multi-task learning in deep neural networks (Source [11]).

Hard Parameter Sharing. The first one, named *hard parameter sharing*, is the most commonly used and was first mentioned in [114]. On this one, we generally share the hidden layers between all tasks while reserving several task-specific output layers (cf. Figure 2.10). This approach reduces the risk of overfitting, which intuitively makes sense since the more tasks we jointly learn, the more the model has to identify representations that satisfy and capture all the tasks, which yields less chance of overfitting the original task [11].

Soft Parameter Sharing. The second way is *soft parameter sharing*, where for each task, we have independent models with their parameters. Part of the models (i.e., module or layers) are regularized to encourage having similar parameters (cf. Figure 2.10).

2.4.2 Mechanisms

Multi-task learning provides better performance due to additional training signals for related tasks. Caruana [109] defined several mechanisms that help multi-task learning generalize better, Ruder [11] further simplified and expanded them. Below, we discuss different examples where we assume we have two related tasks, T and T' relying on hidden layer feature F.

Implicit Data Augmentation. Multi-task learning implicitly augments the data size that we are using for training the model. As all tasks are somewhat noisy, the aim is to learn a good representation for task T that ideally ignores the data-dependent noise and generalizes well when training a model on task T. Different tasks have different noise patterns. A model that learns two tasks simultaneously can learn a more general representation. Learning task T bears the risk of overfitting task T, while learning T and T' enables the model to average noise patterns and better represent F.

Attention Focusing. It is hard for the model to distinguish relevant features from irrelevant ones with limited or noisy data. Here, multi-task learning can support the model to focus only on relevant features since multiple tasks will provide additional evidence for them.

Eavesdropping. In some cases, a number of features are easy to learn through task T' while being challenging to learn through task T. This might be because task T interacts with the features more complexly, or other features prevent the model's learning ability. Multi-task learning allows the model to *eavesdrop*, i.e., learn these features through task T'. This is usually done by directly training the model to predict the features.

Representation Bias. Multi-task learning forces the model to favor representations that are beneficial for other tasks also. It further helps the model to better generalize in new future tasks.

Regularization. As the last one, multi-task learning serves as a regularizer by introducing an inductive bias. In this way, it reduces the risk of overfitting and its ability to fit random noise.

For the models presented in this thesis [14, 15, 18, 27], we employ *multi-task learning* via *hard parameter sharing* where a Transformer-based encoder architecture is simultaneously trained to generate representations that are shared across all the tasks.

CHAPTER 3

Extending Question Answering Resources to Support Answer Verbalization

Answer verbalization task enables QA systems to express and present more natural responses to users while engaging them in a conversation. As a relatively new task, answer verbalization has not been extensively explored with deep learning approaches to observe the possibilities and limits. With the advancement of the deep learning field, the demand for resources has significantly grown. A first step to approach the task with deep learning architectures would be to develop resources that allow training such models. Existing answer verbalization resources [24, 25] comprise only one natural language answer per question, and they also provide the first set of experiments with prominent deep neural models. Considering these works, we observe that a single verbalized answer might not be sufficient for the models to learn diverse patterns for generating the responses. Also, we can think of it from the human point of view, where, given a question, multiple individuals will formulate the answer in different ways. Responses will vary from person to person, but they retain the same meaning.

In this chapter, we address the lack of answer verbalization resources in the KGQA community. More importantly, we want to investigate whether multiple paraphrased answers can improve the performance of machine learning models. The primary objective is to build a new resource upon existing work by generating numerous responses for each question.

We address the following research question in this chapter:

RQ1: How do multiple paraphrased answers affect the performance of answer verbalization?

Contributions of this chapter are summarized as follows:

- We provide a semi-automated framework for generating multiple paraphrase responses for each question using techniques such as back-translation.
- We present the first question answering dataset with multiple paraphrased responses. In particular, the dataset consists of up to eight unique paraphrased responses for each dataset question that can be answered using DBpedia as the underlying KG.
- We provide evaluation baselines that serve to determine our dataset's quality and define a benchmark for future research.

- We analyze the performance of various models when trained with one or more paraphrased answers.

This chapter is based on the following publication ([26]):

- **Endri Kacupaj**, Barshana Banerjee, Kuldeep Singh, and Jens Lehmann. "ParaQA: A Question Answering Dataset with Paraphrase Responses for Single-Turn Conversation." In European Semantic Web Conference, pp. 598-613. Springer, Cham, 2021. DOI: 10.1007/978-3-030-77385-4_36

The rest of the chapter is structured as follows: Section 3.1 introduces the work. In Section 3.2, we describe the related work. We introduce the details of our dataset and the generation workflow in Section 3.3. Section 3.4 describes the availability of the dataset, followed by the experiments in Section 3.5. Section 3.6 provides the experiment results. The reusability study and impact is reported in Section 3.7. Finally, Section 3.8 provides the summary.

3.1 Introduction

In dialog systems research, we can distinguish between single-turn and multi-turn conversations [115, 116]. In single-turn conversations, a user provides all the required information (e.g., slots/values) at once, in one utterance. Conversely, a multi-turn conversation involves anaphora and ellipses to fetch more information from the user as an additional conversation context. The existing ConvQA [16, 117] datasets provide multi-turn dialogues for question answering. In a real-world setting, user will not always require multi-turn dialogues. Therefore, single-turn conversation is a common phenomenon in voice assistants[1]. Some public datasets focus on paraphrasing the questions to provide real-world settings, such as LC-QuAD 2.0 [118] and ComQA [119]. The existing single-turn KGQA datasets provide only up to one verbalization of the response (cf. Table 3.1). In both dataset categories (single-turn or multi-turn), we are not aware of any dataset providing paraphrases of the various answer utterances. For instance, given the question "How many shows does HBO have?", on a KGQA dataset (LC-QuAD [120]), we only find the entity as an answer (e.g. "38"). While on a verbalized KGQA dataset [24], the answer is verbalized as "There are 38 television shows owned by HBO." Given this context, the user can better verify that the system is indeed retrieving the total number of shows owned by HBO. However, the answer can be formulated differently using various paraphrases such as "There are 38 TV shows whose owner is HBO.", "There are 38 television programs owned by that organization" with the same semantic meaning. Hence, paraphrasing the answers can introduce more flexibility and intuitiveness in the conversations. Here, we argue that multiple paraphrased responses improve the machine learning models' performance for answer verbalization on standard empirical metrics.

In this chapter, we introduce ParaQA, a question-answering dataset with multiple paraphrase responses for KGQA. The ParaQA dataset was built using a semi-automated framework that employs advanced paraphrasing techniques such as back-translation. The dataset contains a minimum of two and a maximum of eight unique paraphrased responses per question. We supplement the dataset with several evaluation settings to measure the effectiveness of having multiple paraphrased answers.

[1] `https://docs.microsoft.com/en-us/cortana/skills/mva31-understanding-conversations`

Dataset	Large scale(>=5K)	Complex Questions	SPARQL	Verbalized Answer	Paraphrased Answer
ParaQA (This work)	✓	✓	✓	✓	✓
Free917 [121]	✗	✓	✗	✗	✗
WebQuestions [122]	✓	✗	✗	✗	✗
SimpleQuestions [57]	✓	✗	✓	✗	✗
QALD (1-9)[2]	✗	✓	✓	✗	✗
LC-QuAD 1.0 [120]	✓	✓	✓	✗	✗
LC-QuAD 2.0 [118]	✓	✓	✓	✗	✗
ComplexQuestions [123]	✗	✓	✗	✗	✗
ComQA [119]	✓	✓	✗	✗	✗
GraphQuestions [124]	✓	✓	✓	✗	✗
ComplexWebQuestions [125]	✓	✓	✓	✗	✗
VQuAnDa [24]	✓	✓	✓	✓	✗
CSQA [117]	✓	✓	✗	✗	✗
ConvQuestions [16]	✓	✓	✗	✗	✗

Table 3.1: Comparison of ParaQA with existing QA datasets over various dimensions. Lack of paraphrased utterances of answers remains a key gap in literature.

3.2 Related Work

Our work lies at the intersection of single-turn KGQA and conversational QA datasets. We describe previous efforts and refer to different dataset construction techniques.

KGQA Datasets

The datasets such as SimpleQuestions [57], WebQuestions [126], and the QALD challenge[3] have been inspirational for the evolution of the field. SimpleQuestions [57] dataset is one of the most commonly used large-scale benchmarks for studying single-relation factoid questions over Freebase [127]. LC-QuAD 1.0 [120] was the first large-scale dataset providing complex questions and their SPARQL queries over DBpedia. The dataset has been created using pre-defined templates and a peer-reviewed process to improve those templates. Other datasets such as ComQA [119] and LC-QuAD 2.0 [118] are large-scale QA datasets with complex paraphrased questions without verbalized answers. It is important to note that the answers of most KGQA datasets are non-verbalized. VQuAnDa [24] is the only QA dataset with complex questions containing a single verbalized answer for each question.

Conversational QA Datasets

There has been extensive research for single-turn and multi-turn conversations for open-domain [115, 128, 129]. The research community has recently shifted focus to provide multi-turn conversation datasets for question answering over KGs. CSQA [117] is a large-scale dataset consisting of multi-turn conversations over linked QA pairs. The dataset contained 200K dialogues with 1.6M turns and was collected through a manually intensive semi-automated process. The dataset comprises complex questions that require logical, quantitative, and comparative reasoning over Wikidata KG. ConvQuestions [16] is a crowdsourced benchmark with 11K distinct multi-turn conversations from five different domains ("*Books*", "*Movies*", "*Soccer*", "*Music*", and "*TV Series*"). While both datasets cover multi-turn conversations, none of them contains verbalized answers. Hence, there is a clear gap

[3] http://qald.aksw.org/

in the literature for the datasets focusing on single-turn/multi-turn conversations involving question answering over KGs. In this chapter, we focus on single-turn KGQA to provide ParaQA with multiple paraphrased answers for more expressive conversations.

Dataset Construction Techniques

While some KGQA datasets are automatically generated [130], most of them are manually created either by (i) using in-house workers [120] or crowd-sourcing [118], (ii) or extract questions from online question answering platforms such as search engines, online forum, etc [122]. Most (single-turn/multi-turn) conversational QA datasets are generated using semi-automated approaches [16, 117]. First, conversations are created through predefined templates. Second, the automatically generated conversations are polished by in-house workers or crowd-sourcing techniques. CSQA [117] dataset contains a series of linked QA pairs forming a coherent conversation. Further, these questions are answerable from a KG using logical, comparative, and quantitative reasoning. For generating the dataset, authors first asked pairs of in-house workers to converse with each other. One annotator in a pair acted as a user whose job was to ask questions, and the other annotator worked as the system whose job was to answer the questions or ask for clarifications if required. The annotators' results were abstracted to templates and used to instantiate more questions involving different relations, subjects, and objects. The same process was repeated for different question types such as co-references and ellipses. ConvQuestions [16] dataset was created by posing the conversation generation task on Amazon Mechanical Turk (AMT)[4]. Each crowd worker was asked to build a conversation by asking five sequential questions starting from any seed entity of his/her choice. Humans may have an intuitive model when satisfying their real information needs via their search assistants. Crowd workers were also asked to provide paraphrases for each question. Similar to [117], the crowd workers' results were abstracted to templates and used to create more examples. While both conversational QA datasets use a relatively similar construction approach, none of them considers verbalizing the answers and providing paraphrases for them.

Paraphrasing

In the early years, various traditional techniques have been developed to solve the paraphrase generation problem. McKeown [131] makes use of manually defined rules. Quirk et al. [132] train Statistical Machine Translation (SMT) tools on a large number of sentence pairs collected from newspapers. Wubben et al. [133] propose a phrase-based SMT model trained on aligned news headlines. Recent approaches perform neural paraphrase generation, which is often formalized as a sequence to sequence (seq2seq) learning. Prakash et al. [134] employ a stacked residual LSTM network in the seq2seq model to enlarge the model capacity. Hasan et al. [135] incorporate the attention mechanism to generate paraphrases. Work in [136] integrates the Transformer model and recurrent neural network to learn long-range dependencies in the input sequence.

[4] https://www.mturk.com/

Question	What is the television show whose judges is Randy Jackson?
Answer Verbalizations	1) American Idol is the television show with judge Randy Jackson.
	2) The television show whose judge Randy Jackson is American Idol.
	3) The TV show he's a judge on is American Idol.
Question	How many shows does HBO have?
Answer Verbalizations	1) There are 38 television shows owned by HBO.
	2) There are 38 TV shows whose owner is HBO.
	3) There are 38 television shows whose owner is that organisation.
	4) There are 38 television programs owned by that organization.
Question	From which country is Lawrence Okoye's nationality?
Answer Verbalizations	1) Great Britain is the nationality of Lawrence Okoye.
	2) Great Britain is Lawrence Okoye's citizenship.
	3) The nationality of Lawrence Okoye is Great Britain.
	4) Lawrence Okoye is a Great British citizen.
	5) Lawrence Okoye's nationality is Great Britain.
Question	Does Sonny Bill Williams belong in the Canterbury Bankstown Bulldogs club?
Answer Verbalizations	1) Yes, Canterbury-Bankstown Bulldogs is the club of Sonny Bill Williams.
	2) Yes, the Canterbury-Bankstown Bulldogs is Bill Williams's club.
	3) Yes, the Canterbury-Bankstown Bulldogs is his club.
	4) Yes, Canterbury-Bankstown Bulldogs is the club of the person.
	5) Yes, the club of Sonny Bill Williams is Canterbury-Bankstown Bulldogs.
	6) Yes, Bill Williams's club is the Canterbury-Bankstone Bulldogs.

Table 3.2: Examples from ParaQA.

3.3 ParaQA: A Question Answering Dataset with Paraphrase Responses

The inspiration for generating paraphrased answers originates from the need to provide a context of the question to assure that the query was correctly understood. In that way, the user would verify that the received answer correlates with the question.

For illustration, in our dataset, the question "What is the commonplace of study for jack McGregor and Philip W. Pillsbury?" is translated to the corresponding SPARQL query, which retrieves the result "Yale University" from the KG. In this case, a full natural language response of the result is "Yale University is the study place of Jack McGregor and Philip W. Pillsbury.". As we can see, this form of answer provides us with the query result and details about the query's intention. At the same time, we also provide alternative paraphrased responses such as, "Yale University is the place where both Jack McGregor and Philip W. Pillsbury studied.", "Yale is where both of them studied.". All those responses affirm that the QA system thoroughly comprehended the question context and provided a related answer. The user can further verify that the system retrieves a place where "Jack McGregor" and "Philip W. Pillsbury" went for their studies. Table 3.2 illustrates examples from our dataset.

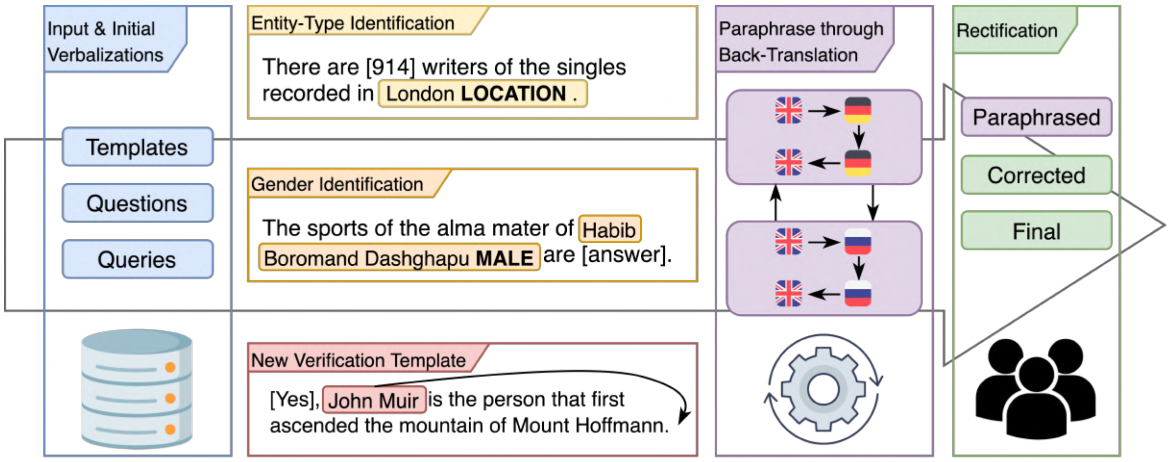
Input & Initial Verbalizations
Templates
Questions
Queries
Entity-Type Identification
There are [914] writers of the singles recorded in London LOCATION .
Gender Identification
The sports of the alma mater of Habib Boromand Dashghapu MALE are [answer].
New Verification Template
[Yes], John Muir is the person that first ascended the mountain of Mount Hoffmann.
Paraphrase through Back-Translation
Rectification
Paraphrased
Corrected
Final

Figure 3.1: Overview of dataset generation workflow. Our proposed generation workflow consists of six modules in total. The first module is "Input & Initial Verbalization", which is responsible for producing the initial verbalized results for each input question. The next three modules ("Entity-Type Identification though Named Entity Recognition", "Gender Identification", and "New Verification Template") are applied simultaneously and provide new verbalized sentences based on the initial ones. Subsequently, the paraphrasing module, named "Paraphrase through Back-Translation", applies back translation to generated answers. Finally, in the last step ("Rectify Verbalization"), we rectify all the paraphrased results through a peer-review process.

3.3.1 Generation Workflow

For generating ParaQA, we decided not to reinvent the wheel to create new questions. Hence, we inherit questions from LC-QuAD [120] and single answer verbalization of these question provided by VQuAnDa [24]. We followed a semi-automated approach to generate the dataset. The overall architecture of the approach is depicted in Figure 3.1.

Input & Initial Verbalization

Our framework requires at least one available verbalized answer per question to build upon it and extend it into multiple diverse paraphrased responses. Therefore, the generation workflow from [24] is adopted as a first step and used to generate the initial responses. This step's inputs are the questions, the SPARQL queries, and the hand-crafted natural language answer templates.

Entity-Type Identification though Named Entity Recognition

The Named Entity Recognition (NER) step recognizes and classifies named entities into predefined categories, for instance, persons, organizations, locations, etc. Here, we aim to identify the entity category (or entity-type) and span and replace it with a predefined value in the response. This stage allows us to accomplish general verbalizations since question entities are swapped with their type categories. The whole process is performed in two steps: 1) A pre-trained NER [137] model is employed to locate entities in the initial generated responses. Discovered entities are replaced with their type category such as "*ORG, PRODUCT, LOC, PERSON, GPE*". 2) A predefined dictionary is used to substitute the type categories with different words such as "the organization, the person, the country". Table 3.3 presents a generated example from the entity-type identification step.

Entity-Type	Question	Count the key people of the Clinton Foundation?
	Initial	There are 8 key people in the **Clinton Foundation**.
	Generated	There are 8 key people in the organisation.
Gender	Question	Which planet was first discovered by Johann Gottfried Galle?
	Verbalized Answer	The planet **discovered by Johann Gottfried Galle** is Neptune.
	Generated	The planet he discovered is Neptune.
Verification	Question	Does the River Shannon originate from Dowra?
	Initial	Yes, **Dowra** is the source mountain of **River Shannon**.
	Generated	Yes, River Shannon starts from Dowra.
Paraphrase	Question	Who first ascended a mountain of Cathedral Peak (California)?
	Initial	**The person that first ascended** Cathedral Peak (California) is John Muir.
	Generated (en-de)	The first person to climb Cathedral Peak (California) is John Muir.
	Generated (en-ru)	The person who first climbed Mount Katty Peak (California) is John Muir.

Table 3.3: Examples generated from each automatic step/module of our proposed generation framework. The presented responses are the outputs from the corresponding modules before they undergo the final peer-review step. The bold text of the initial answer indicates the part of the sentence where the corresponding module is focusing. The underlined text on the generated results reveals the changes made from the module.

Gender Identification

In this step, we create new responses by replacing the question entities with their corresponding pronouns, e.g. "he, she, him, her". This is done by identifying the entity's gender. In particular, we query the KG with a predefined SPARQL query that extracts the gender of the given entity. Based on the position of the entity in the answer, we replace it with the appropriate pronoun. Table 3.3 illustrates a generated example from the gender identification step. In the peer-review process, we verify the dataset to avoid bias in the genders, considering we extract gender information from DBpedia, and sometimes KG data quality is not perfect.

New Verification Template

Considering that, on verification questions, all triple data is given (subject, predicate, object). We introduce a verbalization template that interchanges the head and tail triple information and generate more diverse responses. Table 3.3 provides a generated example from this process.

Paraphrase through Back-Translation

After having assembled sufficient answers for each question, we employ a paraphrasing strategy through a back-translation approach. In general, back-translation is when a translator (or team of translators) interprets or re-translate a document that was previously translated into another language back to the original language. In our case, the two translators are independent models, and the second model has no knowledge or contact with the original text.

In particular, inspired by [138, 139], our initial responses alongside the new proposed answer templates are paraphrased using Transformer-based models [82] as translators. The model is evaluated successfully on the WMT'18[5] dataset that includes translations between different languages. In our case, we perform back-translation with two different sets of languages: 1) Two Transformer models are used to translate the responses between English and German language (en→de→en). 2) Another two models are used to translate between English and Russian language (en→ru→en). Here it is worth mentioning that we also forwarded output responses from one translation stack into the other (e.g., en→de→en→ru→en). In this way, we generate as many as possible different paraphrased responses. Please note that the selection of languages for back translation was done considering our inherited underlying model's accuracy in machine translation tasks on WMT'18. Table 3.3 illustrates some examples from our back-translation approach.

Rectify Verbalization

After collecting multiple paraphrased versions of the initial responses, the last step is to rectify and rephrase them to sound more natural and fluent. The rectification step of our framework is done through a peer-review process to ensure the answers' grammatical correctness. Finally, by the end of this step, we will have at least two and at most eight diverse paraphrased responses per question, including the initial answer.

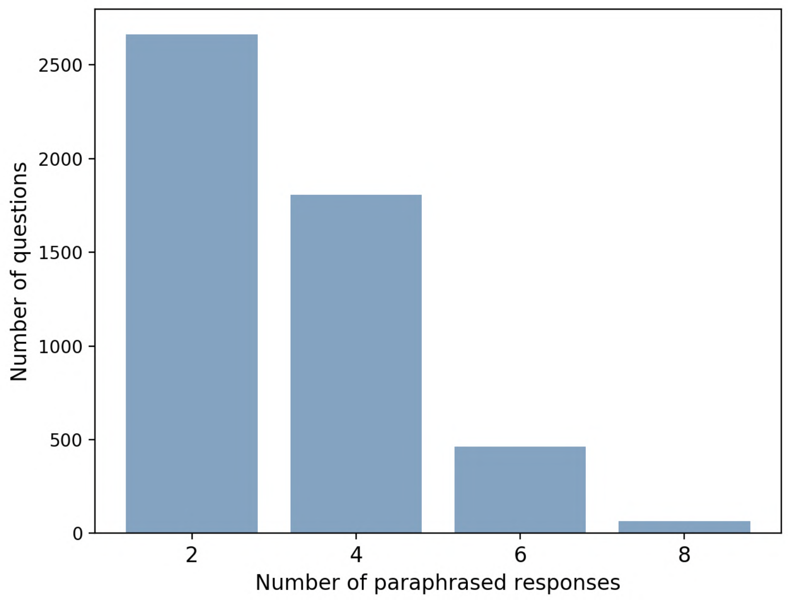

Figure 3.2: Total paraphrased responses per question.

[5] http://www.statmt.org/wmt18/translation-task.html

3.3.2 Dataset Statistics

We provide dataset insights regarding its total paraphrased results for each question and the percentage of generated answers from each module on our framework. Figure 3.2 illustrates the distribution of 5000 questions of ParaQA based on a total number of paraphrased responses per question. As seen from the figure, more than 2500 questions contain at most two paraphrased results. A bit less than 2000 questions include at most four answers, while around 500 have no less than six paraphrased answers. Finally, less than 100 examples contain at most eight paraphrased results. Figure 3.3 depicts the percentage of generated answers for each step from our generation workflow. The first step (input and initial verbalization) provides approximately 30% of our total results, while the next three steps (entity type identification, gender identification, and new verification templates) produce roughly 20% of responses. Finally, the back-translation module generates no less than 50% of the complete paraphrased answers in ParaQA.

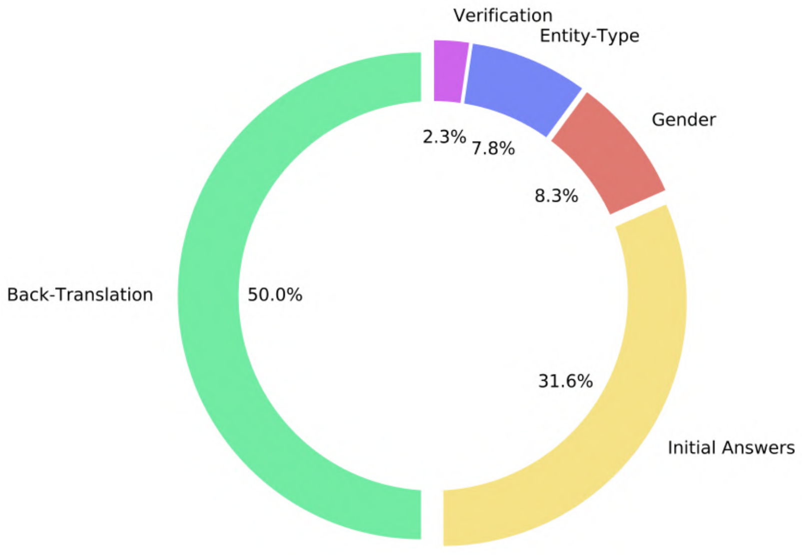

Figure 3.3: Percentage of generated results from each step.

3.4 Availability and Sustainability

The dataset is available at a GitHub repository[6] under the Attribution 4.0 International (CC BY 4.0)[7] license. As a permanent URL, we also provide our dataset through figshare at https://figshare.com/projects/ParaQA/94010. The generation framework is also available at a GitHub repository[8] under the MIT License[9]. Please note, the dataset and the experiments reported in the chapter are in two different repositories due to the free distributed license agreement.

[6] https://github.com/barshana-banerjee/ParaQA
[7] https://creativecommons.org/licenses/by/4.0/
[8] https://github.com/barshana-banerjee/ParaQA_Experiments
[9] https://opensource.org/licenses/MIT

The maintenance is ensured through the CLEOPATRA[10] project till 2022. After that, the maintenance of the resource will be handled by the question and answering team of the Smart Data Analytics (SDA)[11] research group at the University of Bonn and at Fraunhofer IAIS[12].

3.5 Experimental Setup

To assure the quality of the dataset and the advantage of having multiple paraphrased responses, we perform experiments and provide baseline models, which researchers can use as a reference point for future research.

Baseline Models

For the baselines, we employ three sequence to sequence models. Sequence to sequence is a family of machine learning approaches used for language processing and often used for natural language generation tasks. The first model consists of an RNN [71] based architecture, the second uses a convolutional network [140], while the third employs a Transformer network [82].

Evaluation Metrics

BLEU (Bilingual Evaluation Understudy). BLEU score introduced by [141] is so far the most popularly used machine translation metric to evaluate the quality of the model generated text compared to human translation. It aims to count the n-gram overlaps in the reference by taking the maximum count of each n-gram, and it clips the count of the n-grams in the candidate translation to the maximum count in the reference. Essentially, BLEU is a modified version of precision to compare a candidate with a reference. However, candidates with a shorter length than the reference tend to give a higher score, while the modified n-gram precision already penalizes longer candidates. Brevity penalty (BP) was introduced to rectify this issue and defined as:

$$BP = \begin{cases} 1, & c \geq r, \\ exp(1 - \frac{r}{c}), & c < r. \end{cases} \tag{3.1}$$

Where it gets the value of 1 if the candidate length c is larger or equal to the reference length r. Otherwise, is set to $\exp(1 - r/c)$. Finally, a set of positive weights $\{w_1, ..., w_N\}$ is determined to compute the geometric mean of the modified n-gram precision. The BLEU score is calculated by:

$$BLEU = BP \cdot exp(\sum_{n=1}^{N} w_n log(P_n)), \tag{3.2}$$

where N is the number of different n-grams. In our experiments, we employ $N = 4$ (which is a default value) and uniform weights $w_n = 1/N$.

[10] http://cleopatra-project.eu/
[11] https://sda.tech/
[12] https://www.iais.fraunhofer.de/

METEOR (Metric for Evaluation of Translation with Explicit ORdering). METEOR score, introduced by [142], is a metric for the evaluation of machine-translation output. METEOR is based on the harmonic mean of unigram precision and recall, with recall weighted higher than precision.

BLEU score suffers from the issue that the BP value uses lengths that are averaged over the entire corpus level, leading to having individual sentences a hit. In contrast, METEOR modifies the precision at sentence or segment level, replacing them with a weighted F-score based on mapping uni-grams and a penalty function that solves the existing problem. Similar to BLEU, METEOR score can be in the range of 0.0 and 100, with 100 being the best score. Formally we define it as:

$$\begin{aligned} F_{mean} &= \frac{P \cdot R}{\alpha \cdot P + (1-\alpha) \cdot R}, \\ Pen &= \gamma \cdot \left(\frac{ch}{m}\right)^{\beta}, \\ METEOR &= (1 - Pen) \cdot F_{mean} \end{aligned} \tag{3.3}$$

where P and R are the uni-gram precision and recall respectively, and are used to compute the parametrized harmonic mean F_{mean}. Pen is the penalty value and is calculated using the counts of chunks ch and the matches m. α, β and γ are free parameters used to calculate the final score. For our experiments we employ the common values of $\alpha = 0.9$, $\beta = 3.0$ and $\gamma = 0.5$.

Training and Configurations

The experiments are performed to test how easy it is for a standard sequence to sequence model to generate the verbalized response using as input only the question or the SPARQL query. Inspired by [24], during our experiments, we prefer to hide the query answer from the responses by replacing it with a general answer token. In this way, we simplify the model task to predict only the query answer's position in the final verbalized response.

Furthermore, we perform experiments with three different dataset settings. We intend to illustrate the advantage of having multiple paraphrased responses compared to one or even two. Therefore, we run individual experiments by using one response, two responses, and finally, multiple paraphrased responses per question. To conduct the experiments for the last two settings, we forward the responses associated with their question into our model. We calculate the scores for each generated response by comparing them with all the existing references. For the sake of simplicity, and as done by [143], the final score is the maximum value achieved for each generated response.

For fair comparison across the models, we employ similar hyperparameters for all. We utilize an embeddings dimension of 512, and all models consist of 2 layers. We apply dropout with probability 0.1. We use a batch size of 128, and we train for 50 epochs. Across all experiments, we use Adam optimizer and cross-entropy as a loss function. To facilitate reproducibility and reuse, our baseline implementations and results are publicly available[13].

3.6 Results

Table 3.4 and Table 3.5 illustrate the experiment results for BLEU and METEOR scores, respectively. For both metrics, the convolutional model performs the best. It outperforms the RNN and Transformer models in different inputs and responses. Here, it is more interesting to notice that all models perform better with multiple paraphrased answers than one or two responses. At the same time, the scores with two answers are better than those with a single response. Hence, we can assume that the more

[13] https://github.com/barshana-banerjee/ParaQA_Experiments

Model	Input	One Response	Two Responses	Multiple Paraphrased
RNN [71]	Question	15.43	18.8	22.4
	SPARQL	20.1	21.33	26.3
Transformer [82]	Question	18.3	21.2	23.6
	SPARQL	23.1	24.7	28.0
Convolutional [140]	Question	21.3	25.1	**25.9**
	SPARQL	26.02	28.4	**31.8**

Table 3.4: BLEU score experiment results.

Model	Input	One Response	Two Responses	Multiple Paraphrased
RNN [71]	Question	53.1	56.2	58.4
	SPARQL	57.0	59.3	61.8
Transformer [82]	Question	56.8	58.8	59.6
	SPARQL	60.1	63.0	63.7
Convolutional [140]	Question	57.5	58.4	**60.8**
	SPARQL	64.3	65.1	**65.8**

Table 3.5: METEOR score experiment results.

paraphrased responses we have, the better the model performance. Concerning the experiment inputs (Question, SPARQL), as indicated by both metrics, we obtain improved results with SPARQL on all models and responses. As expected, this is due to the constant input pattern templates that the SPARQL queries have. While with questions, we end up having a different reworded version for the same template. We expect the research community to use these models as baselines to develop more advanced approaches targeting either single-turn conversations for QA or answer verbalization.

3.7 Reusability and Impact

ParaQA dataset can fit in different research areas. Undoubtedly, the most suitable one is in the single-turn question answering over KGs for supporting a more expressive QA experience. The dataset offers the opportunity to build end-to-end machine learning frameworks to handle both tasks of query construction and natural language response generation. Simultaneously, the dataset remains useful for any QA sub-task, such as entity/relation recognition, linking, and disambiguation.

Besides the QA research area, the dataset is also suitable for Natural Language Generation (NLG) tasks. As we accomplish in our experiments, using as input the question or the query, the NLG task will generate the best possible response. We also find ParaQA suitable for the NLP area of paraphrasing. Since we provide more than one paraphrased example for each answer, researchers can experiment with the dataset for building paraphrasing systems for short texts. Furthermore, our dataset can also be used for the research involving SPARQL verbalization, which has been a long-studied topic in the Semantic Web community [22, 23, 144].

3.8 Summary

In this chapter, we present ParaQA – the first single-turn question answering dataset with multiple paraphrased responses. Alongside the dataset, we provide a semi-automated framework for generating various paraphrase responses using back-translation techniques. Finally, we also share a set of evaluation baselines and illustrate the advantage of multiple paraphrased answers through commonly used metrics such as BLEU and METEOR. Evaluation results indicate improved performance when employing more than one verbalized answer. The dataset offers a worthwhile contribution to the community, providing the foundation for numerous research lines in the single-turn KGQA domain and others.

CHAPTER 4

Answer Verbalization via Multi-Task Learning

In Chapter 3, we focused on generating a KGQA resource with multiple paraphrased answers. At the same time, we investigated the advantage of having multiple answers instead of one by hypothesizing the positive impact on machine learning models. Existing answer verbalization approaches involve sequence to sequence methods that employ only the input question to generate natural language answers. However, relying only on one source of information for verbalizing answers can often yield wrong results due to the varying format of the questions. A source of information with more consistent patterns will allow the model to be more accurate in verbalizing answers. In the previous chapter's experiments, we noticed that providing the queries instead of the question as input produces better results due to the constant template patterns that the queries have.

In an attempt to improve the performance of answer verbalization, we dedicate this chapter to setting up a framework that incorporates logical forms as additional context. The framework contains several sub-tasks which we coordinate and train via multi-task learning. For ascertaining the novelty of the proposed approach and to encourage a broader usage, we provide an evaluation where we extend a basic QA system with the answer verbalization task.

We address the following research question in this chapter:

RQ2: How can we incorporate logical forms to improve the performance of answer verbalization via multi-task learning?

Contributions of this chapter are summarized as follows:

- We introduce a multi-task-based hybrid answer verbalization framework that consists of four modules trained simultaneously.
- We propose a similarity threshold and cross attention modules to determine the relevance between the inputs and fuse information to employ a hybrid strategy.
- We provide an extensive evaluation and ablation study of the proposed framework on three QA datasets with answer verbalization. Our evaluation results establish a new baseline for the answer verbalization task, which we believe will drive future research in a newly studied problem.

This chapter is based on the following publication ([26]):

- **Endri Kacupaj**, Shyamnath Premnadh, Kuldeep Singh, Jens Lehmann, and Maria Maleshkova. "VOGUE: Answer Verbalization Through Multi-Task Learning." In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 563-579. Springer, Cham, 2021. DOI: 10.1007/978-3-030-86523-8_34

The structure of the chapter is as follows: Section 4.1 introduces the work. Section 4.2 presents the related work. Section 4.3 provides the task definition. Section 4.4 presents the proposed framework. Section 4.5 describes the multi-task learning process. Section 4.6 describes the experimental setup, while Section 4.7 the experiments results. Section 4.8 provides a detailed ablation study. An error analysis is on Section 4.9 and a case study on Section 4.10. We summarize in Section 4.11.

4.1 Introduction

Existing open-source KGQA systems are restricted to only generating or producing answers without verbalizing them in natural language [145, 146]. The lack of verbalization makes the interaction with the user not natural in contrast to voice assistants such as Siri and Alexa. Figure 4.1 depicts an ideal integration of a QA pipeline with answer verbalization. For instance, assuming that the answer to the exemplary question, "How many shows does HBO have?" is not known by the user. Suppose the QA system only responds with a number (e.g., 38) as an answer (similar to open-source KGQA systems), with no further explanation. In that case, the user might need to refer to an external data source to verify the answer. In an effort to enable the users to verify the answer provided by a QA system, researchers employed techniques such as (i) revealing the generated formal query [20], (ii) graphical visualizations of the formal query [19] and (iii) verbalizing the formal query [22]. Understanding the necessity of verbalized answers in the KGQA domain, recently, several datasets have been proposed [24, 25]. For the answer verbalization task, the system has to verbalize the answer to convey not only the information requested by the user but also additional characteristics that indicate how the answer was determined. In our exemplary question (from dataset [26]), a verbalized response would look like, "HBO owns 38 television shows." or "There are 38 TV shows whose owner is HBO.". Both answers allow the user to verify that the system retrieved the total number of TV shows owned by HBO. In the literature, there exist empirical results showing that answer verbalization quantitatively and qualitatively improves the ability to understand the answer [6, 24, 26]. Furthermore, in the previous chapter, we saw how multiple answers could positively impact the models' performance by running experiments with the question or the query (logical form) as inputs. However, it remains an open challenge – how can we verbalize an answer, given a logical form and an input question. In this chapter, we address precisely this open and highly relevant research challenge with our work.

We propose VOGUE (**V**erbalization thr**O**u**G**h m**U**lti-task l**E**arning), the first approach dedicated to verbalize answers for KGQA. Our idea is to employ the question (user utterance) and the QA system-generated query as inputs. We refer to this strategy as "hybrid", since the final verbalized answer is produced using both the question and query concurrently. This work argues that leveraging content from both sources allows the model for better convergence and provides new, improved results. Furthermore, we complement our hypothesis by utilizing a multi-task learning paradigm since it has been quite efficient for different system architectures [147], including question answering systems [14, 15].

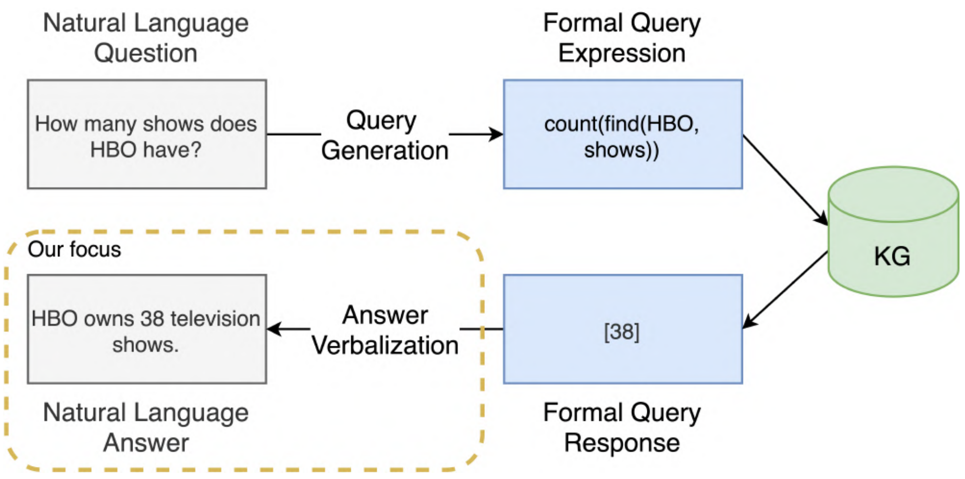

Figure 4.1: A QA pipeline with integrated answer verbalization module. Our focus is the answer verbalization task as we assume logical form is generated by a QA system using the input question.

Our proposed framework can receive two (e.g., question & query) or even one (e.g., question) input. It consists of four modules that are trained simultaneously to generate the verbalized answer. The first module employs a dual Transformer-based encoder architecture for encoding the inputs. The second module determines whether the encoded inputs are relevant and decides if both will be used for verbalization. The third module consists of a cross-attention network that performs question and query matching by jointly modeling the relationships of question words and query actions. Finally, the last module employs a Transformer decoder that is used to generate the final verbalization. To facilitate reproducibility and reuse, our framework implementation is publicly available[1].

4.2 Related Work

As part of the related work we describe previous efforts and refer to different approaches from research fields, including task-oriented dialog systems, WebNLG, and KGQA systems.

Task-oriented Dialog Systems

A task-oriented dialogue system aims to help the user complete certain tasks in a specific domain (e.g. restaurant booking, weather query, or flight booking), making it valuable for real-world business. Typically, task-oriented dialogue systems are built on top of a structured ontology, which defines the tasks' domain knowledge. Wen et al. [148] proposed a modular architecture in which each component is formed of neural networks, making the model end-to-end differentiable. Bordes et al. [149] formalized the task-oriented dialogue as a reading comprehension task regarding the dialogue history as context, user utterance as the question, and system response as the answer. In their work, authors utilized end-to-end memory networks for multi-turn inference. Madotto et al. [150] took a similar approach and further feed the knowledge base information into the memory networks. Eric et al. [151] introduced a new memory network structure named key-value memory networks to extract

[1] https://github.com/endrikacupaj/VOGUE

relevant information from KB through key-value retrieval. In [152], authors proposed a two-step seq2seq generation model, which bypassed the structured dialogue act representation and only retain the dialogue state representation. In their method, the model first encodes the dialogue history and then generates a dialogue state using LSTM and CopyNet [153]. Given the state, the model then generates the final natural language response. Kassawat et al. [154] proposed RNN-based end-to-end encoder-decoder architecture, which employs joint embeddings of the knowledge graph and the corpus as input. The model provides an additional integration of user intent and text generation, trained through a multi-task learning paradigm.

WebNLG

The WebNLG is a challenge that consists of mapping structured data to a textual representation. The dataset [155] contains data/text pairs where the data is a set of triples extracted from DBpedia, and the text is the verbalization of these triples. The dataset has been promoted for the development of (i) RDF verbalizers and (ii) microplanners to handle a wide range of linguistic constructions. In our case, we focus on related works that concentrate on RDF verbalizers. Gao et al. [156] proposes an RDF-to-text model that jointly learn local and global structure information via combining two graph-augmented structural neural encoders for the input triples. Zhao et al. [157] propose DualEnc, a dual encoding model that can incorporate the graph structure and cater to the linear structure of the output text. Song et al. [158] proposes a graph-to-text approach that leverages richer training signals to guide the model for preserving input information. They introduce two types of autoencoding losses, each individually focusing on different aspects of input graphs. The losses are then back-propagated to calibrate the model via multi-task training. Liu et al. [159] propose an attention-based model, which mainly contains an entity extraction module and a relation detection module. The model devises a supervised multi-head self-attention mechanism as the relation detection module to learn the token-level correlation for each relation type separately. Shen et al. [160] propose an approach to explicitly segment target text into fragment units and aligning them with their data correspondences. The segmentation and correspondence are jointly learned as latent variables without any human annotations. They further impose a soft statistical constraint to regularize the segmental granularity. Ngonga Ngomo et al. [23] present SPARQL2NL, a generic approach that allows verbalizing SPARQL queries; besides, the approach can describe the output of queries by providing a natural-language description that led to the result. Work from [21] present SPARQLtoUser, a multilingual method to produce a user understandable version of a SPARQL that can operate on multiple knowledge bases.

The fields of task-oriented dialogue systems and WebNLG contain various approaches for generating text; nevertheless, none of them can be applied directly to solve answer verbalization for KGQA systems. Most task-oriented dialogue systems are designed and implemented to fit their corresponding task, and therefore they would not be suitable for open-domain knowledge graphs (e.g. Wikidata, DBpedia). Regarding WebNLG, the task only considers triples or graph structure data as input. In answer verbalization, the model input can be the question and/or the query. While the query can be translated into a graph structure, there is no support for textual information such as the question.

4.3 Task Definition

In this work, we target the problem of answer verbalization for KGQA. A semantic parsing-based QA system maps a question to a logical form and executes it on a KG to produce the answer. For our task, given the question, the generated logical form, and the extracted answer, we aim to generate a natural language sentence, with the requirements that it is grammatically sound and correctly represents all the information in the question, logical form, and answer. Formally, let X, Y denote the source-target pair. X contains the set of questions, logical forms, answers, and Y corresponds to $y_1, y_2, ..., y_m$, which is the verbalized answer of X. The goal of the answer verbalization is to learn a distribution $p(Y|X)$ to generate natural language text describing the answer automatically.

4.4 Verbalization Through Multi-Task Learning

In question answering, the input data consists of question q and its answer a, extracted from the knowledge graph. The QA system will map the question to a logical form l depending on the context. For answer verbalization, VOGUE maps the question, logical form, and answer to natural language sentence s. Figure 4.2 shows the architecture of VOGUE.

4.4.1 Dual Encoder

To encode both the question and logical form, we employ a dual encoder architecture. Our dual encoder consists of two instances of the Transformer encoder [82].

First, as a preprocessing, we use a previous competitive pre-trained named entity recognition model [161] to identify and replace all entities in the question with a more general entity token $[ENT]$. In this way, we allow our model to focus on the sentence structure and relations between words. Furthermore, our model learns the positions of entities in the question. It also allows VOGUE to predict the respective entity positions in the verbalized answer. The same preprocessing step applies to the logical form. At the end of each input, we append a context token $[CTX]$, which is used later as a semantic representation.

Next, given the question utterance q containing n words $\{w_1, \ldots, w_n\}$ and the logical form l containing m actions $\{a_1, \ldots, a_m\}$, we tokenize the contexts and use the pre-trained model GloVe [162] to embed the words into a vector representation space of dimension d [2]. Our word embedding model provides us with the sequences $x^{(q)} = \{x_1^{(q)}, \ldots, x_n^{(q)}\}$, $x^{(lf)} = \{x_1^{(lf)}, \ldots, x_m^{(lf)}\}$ where $x_i^{(q)}$, $x_i^{(lf)}$ are given by,

$$\begin{aligned} x_i^{(q)} &= GloVe(w_i), \\ x_i^{(lf)} &= GloVe(a_i), \end{aligned} \tag{4.1}$$

and $x_i^{(q)}, x_i^{(lf)} \in \mathbb{R}^d$. Afterwards, both sequences are forwarded through the Transformer encoders. The two encoders here output the contextual embeddings $h^{(q)} = \{h_1^{(q)}, \ldots, h_n^{(q)}\}$ and $h^{(lf)} = \{h_1^{(lf)}, \ldots, h_m^{(lf)}\}$, where $h_i^{(q)}, h_i^{(lf)} \in \mathbb{R}^d$. We define this as:

$$\begin{aligned} h^{(q)} &= encoder_q(x_q; \theta^{(enc_q)}), \\ h^{(lf)} &= encoder_{lf}(x_{lf}; \theta^{(enc_{lf})}), \end{aligned} \tag{4.2}$$

where $\theta^{(enc_q)}$, $\theta^{(enc_{lf})}$ are the encoders trainable parameters.

[2] We employ the same dimension d for all the representations, unless it is explicitly mentioned.

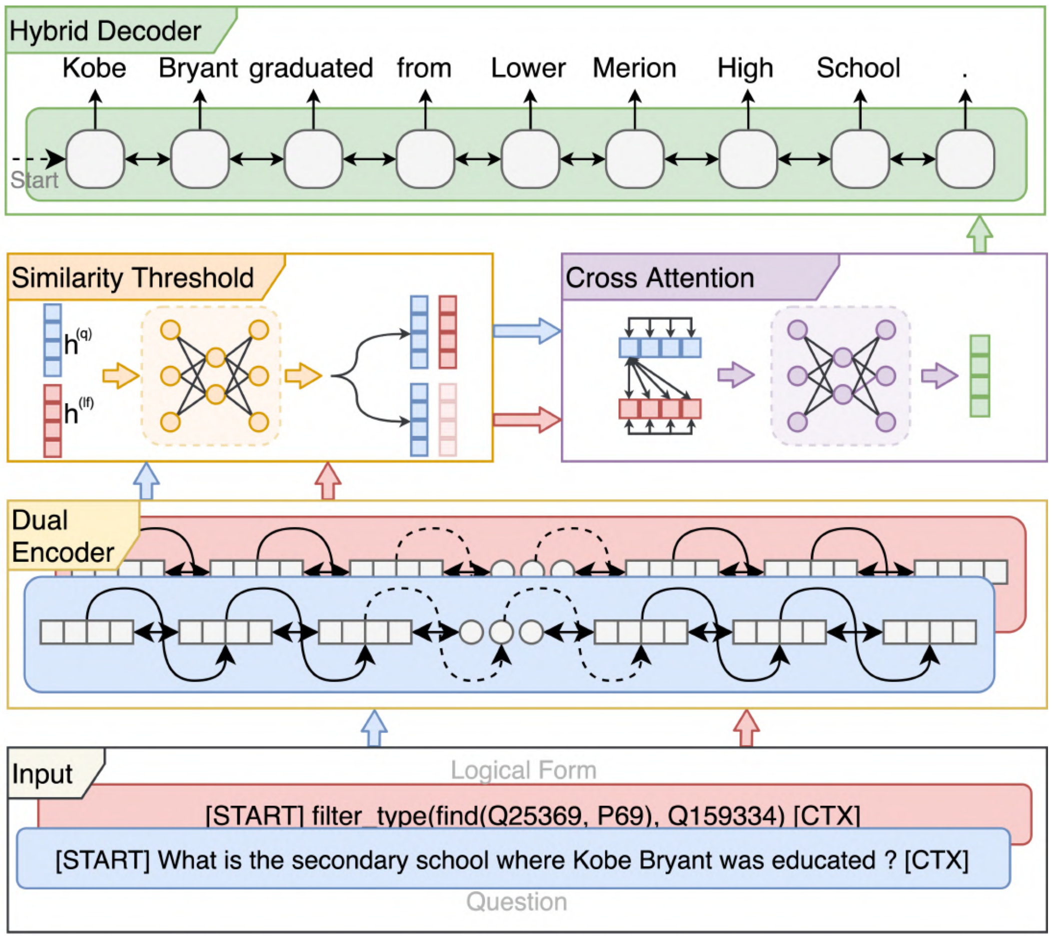

Figure 4.2: VOGUE (**V**erbalization thr**O**u**G**h m**U**lti-task l**E**arning) architecture. It consists of four modules: 1) A dual encoder that is responsible to encode both inputs (question, logical form). 2) A similarity threshold module that determines whether the encoded inputs are relevant and determines if both will be used for verbalization. 3) A cross-attention module that performs question and query matching by jointly modeling the relationships of question words and query actions. 4) A hybrid decoder that generates the verbalized answer using the information of both question and logical form representations from the cross-attention module.

4.4.2 Similarity Threshold

Given the encoded question utterance and logical form, VOGUE's second module is responsible for learning the relevance between the inputs and determining whether we will employ both for verbalization. This module is necessary when we want to utilize our framework alongside a question answering system. If we assume that the QA system is perfect and always produces correct logical forms, this module can be skipped. However, in a real-world scenario, QA systems are far from perfect. Therefore, we employ this module, which intends to identify the threshold for determining if two inputs are similar or not. The input here is the concatenation of the hidden states of the encoded question utterance $h^{(q)}$ and logical form $h^{(lf)}$. The module will perform binary classification on the vocabulary $V^{(st)} = \{0, 1\}$, where 0 indicates that there is no high relevance between the inputs, and only the question will be used for verbalization. While 1 allows us to use both and continue with the next module. Overall, our similarity threshold module is implemented using two linear layers, a Leaky ReLU activation function and a softmax for the predictions. Formally we define the module as:

$$\begin{aligned} h^{(st)} &= LeakyReLU(\boldsymbol{W}^{(st_1)}[h^{(q)}; h^{(lf)}]), \\ p^{(st)} &= softmax(\boldsymbol{W}^{(st_2)} h^{(st)}), \end{aligned} \tag{4.3}$$

where $\boldsymbol{W}^{(st_1)} \in \mathbb{R}^{d \times 2d}$ are the weights of the first linear layer and $h^{(st)}$ is the hidden state of the module. $\boldsymbol{W}^{(st_2)} \in \mathbb{R}^{|V^{(st)}| \times d}$ are the weights of the second linear layer, $|V^{(st)}|$ is the size of the vocabulary and $p^{(st)}$ denotes the probability distribution over the vocabulary indices.

4.4.3 Cross Attention

Inspired by recent computer vision research [163, 164], we employ a cross-attention module that exploits relationships between the inputs and fuses information. The module here performs question and logical form matching by jointly modeling the relationships of question words and logical form actions. Our cross-attention approach is a variation of the self-attention mechanism [82]. In the self-attention mechanism the output is determined by a query and a set of key-value pairs. Given the stacked encoded question and logical form, $h^{(qlf)} = \binom{h^{(q)}}{h^{(lf)}} = \{h_1^{(q)}, \ldots, h_n^{(q)}; h_1^{(lf)}, \ldots, h_m^{(lf)}\}$, where $h^{(qlf)} \in \mathbb{R}^{2 \times d}$ we calculate the query and key-value pairs using three linear projections:

$$\begin{aligned} \boldsymbol{Q}^{(qlf)} &= \boldsymbol{W}^{(Q)} h^{(qlf)} = \begin{pmatrix} \boldsymbol{W}^{(Q)} h^{(q)} \\ \boldsymbol{W}^{(Q)} h^{(lf)} \end{pmatrix} = \begin{pmatrix} \boldsymbol{Q}^{(q)} \\ \boldsymbol{Q}^{(lf)} \end{pmatrix}, \\ \boldsymbol{K}^{(qlf)} &= \boldsymbol{W}^{(K)} h^{(qlf)} = \begin{pmatrix} \boldsymbol{W}^{(K)} h^{(q)} \\ \boldsymbol{W}^{(K)} h^{(lf)} \end{pmatrix} = \begin{pmatrix} \boldsymbol{K}^{(q)} \\ \boldsymbol{K}^{(lf)} \end{pmatrix}, \\ \boldsymbol{V}^{(qlf)} &= \boldsymbol{W}^{(V)} h^{(qlf)} = \begin{pmatrix} \boldsymbol{W}^{(V)} h^{(q)} \\ \boldsymbol{W}^{(V)} h^{(lf)} \end{pmatrix} = \begin{pmatrix} \boldsymbol{V}^{(q)} \\ \boldsymbol{V}^{(lf)} \end{pmatrix}, \end{aligned} \tag{4.4}$$

where $\boldsymbol{W}^{(Q)}, \boldsymbol{W}^{(K)}, \boldsymbol{W}^{(V)} \in \mathbb{R}^{d \times d}$ are the weights of the linear layers and $\boldsymbol{Q}^{(qlf)}, \boldsymbol{K}^{(qlf)}, \boldsymbol{V}^{(qlf)}$ are the query, key and value of the stacked question and logical form. Next, for calculating the cross-attention we simplify the "*Scaled Dot-Product Attention*" [82] step by removing the scaling factor and softmax. We end-up calculating the attention of our input as:

$$\begin{aligned} Attention(\boldsymbol{Q}^{(qlf)}, \boldsymbol{K}^{(qlf)}, \boldsymbol{V}^{(qlf)}) &= \boldsymbol{Q}^{(qlf)} \boldsymbol{K}^{(qlf)T} \cdot \boldsymbol{V}^{(qlf)} \\ &= \begin{pmatrix} \boldsymbol{Q}^{(q)} \\ \boldsymbol{Q}^{(lf)} \end{pmatrix} (\boldsymbol{K}^{(q)T} \boldsymbol{K}^{(lf)T}) \cdot \begin{pmatrix} \boldsymbol{V}^{(q)} \\ \boldsymbol{V}^{(lf)} \end{pmatrix} \\ &= \begin{pmatrix} \boldsymbol{Q}^{(q)} \boldsymbol{K}^{(q)T} & \boldsymbol{Q}^{(q)} \boldsymbol{K}^{(lf)T} \\ \boldsymbol{Q}^{(lf)} \boldsymbol{K}^{(q)T} & \boldsymbol{Q}^{(lf)} \boldsymbol{K}^{(lf)T} \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{V}^{(q)} \\ \boldsymbol{V}^{(lf)} \end{pmatrix} \\ &= \begin{pmatrix} \boldsymbol{Q}^{(q)} \boldsymbol{K}^{(q)T} \boldsymbol{V}^{(q)} + \boldsymbol{Q}^{(q)} \boldsymbol{K}^{(lf)T} \boldsymbol{V}^{(lf)} \\ \boldsymbol{Q}^{(lf)} \boldsymbol{K}^{(lf)T} \boldsymbol{V}^{(lf)} + \boldsymbol{Q}^{(lf)} \boldsymbol{K}^{(q)T} \boldsymbol{V}^{(q)} \end{pmatrix}. \end{aligned} \tag{4.5}$$

When we calculate the cross-attention for the question, we also use the key-value pair from the logical form $(\boldsymbol{K}^{(lf)}, \boldsymbol{V}^{(lf)})$, the same applies when calculating the cross-attention for the logical form. After calculating the cross-attentions, we use the same steps as in the Transformer to produce the new representations for our inputs. Finally, considering $h^{(qca)}$, $h^{(lfca)}$ the output representations of the cross-attention module for the question and logical form respectively, we concatenate them and forward them to the hybrid decoder module.

4.4.4 Hybrid Decoder

To translate the input question and logical form into a sequence of words (verbalized answer), we utilize a Transformer decoder architecture [82], which employs the multi-head attention mechanism. The decoder will generate the final natural language answer. The output here is dependent on the cross-attention embedding $h^{(ca)}$. Here we define the decoder vocabulary as

$$V^{(dec)} = V^{(vt)} \ \cup \ \{ \ [START], \ [END], \ [ENT], \ [ANS] \ \}, \tag{4.6}$$

where $V^{(vt)}$ is the vocabulary with all the distinct tokens from our verbalizations. As we can see, the decoder vocabulary contains four additional helper tokens, where two of them ($[START]$, $[END]$) indicate when the decoding process starts and ends, while the other two ($[ENT]$, $[ANS]$) are used to specify the position of the entities and the answer on the final verbalized sequence. On top of the decoder stack, we employ a linear layer alongside a softmax to calculate each token's probability scores in the vocabulary. We define the decoder stack output as follows:

$$\begin{aligned} h^{(dec)} &= decoder(h^{(ca)}; \theta^{(dec)}), \\ p_t^{(dec)} &= softmax(\boldsymbol{W}^{(dec)} h_t^{(dec)}), \end{aligned} \tag{4.7}$$

where $h_t^{(dec)}$ is the hidden state in time step t, $\theta^{(dec)}$ are the decoder trainable parameters, $\boldsymbol{W}^{(dec)} \in \mathbb{R}^{|V^{(dec)}| \times 2d}$ are the linear layer weights, and $p_t^{(dec)} \in \mathbb{R}^{|V^{(dec)}|}$ is the probability distribution over the decoder vocabulary in time step t. $|V^{(dec)}|$ denotes the decoder's vocabulary size.

4.5 Mutli-Task Learning

The framework consists of four trainable modules. However, we apply a loss function only on two of them (similarity threshold and hybrid decoder). The dual encoder and cross-attention modules are trained based on the similarity threshold and hybrid decoder's signal. To account for multi-tasking, we perform a weighted average of all the single losses:

$$L = \lambda_1 L^{st} + \lambda_2 L^{dec}, \tag{4.8}$$

where λ_1, λ_2 are the relative weights learned during training considering the difference in magnitude between losses by consolidating the log standard deviation [147, 165]. L^{st} and L^{dec} are the respective negative log-likelihood losses of the similarity threshold and hybrid decoder modules. These losses are defined as:

$$\begin{aligned} L^{st} &= -\sum_{j=1}^{2d} logp(y_j^{(st)}|x), \\ L^{dec} &= -\sum_{k=1}^{m} logp(y_k^{(dec)}|x), \end{aligned} \tag{4.9}$$

where m is the length of the gold logical form. $y_j^{(st)} \in V^{(st)}$ are the gold labels for the similarity threshold and $y_k^{(dec)} \in V^{(dec)}$ are the gold labels for the decoder. The model benefits from each module's supervision signals, which improves the performance in the given task.

4.6 Experimental Setup

In this section, we provide the setup for the experiments. We further describe the resources and metrics we employ.

Datasets

We perform experiments on three answer verbalization datasets (cf. Table 4.1). Below we provide a brief description of these:

- VQuAnDa [24] is the first QA dataset, which provides the verbalization of the answer in natural language. It contains 5000 "*complex*" questions with their SPARQL queries and answers verbalization. The dataset consists of 5042 entities and 615 relations.
- ParaQA (Chapter 3) [26] is a QA dataset with multiple paraphrased responses. The dataset was created using a semi-automated framework for generating diverse paraphrasing of the answers using techniques such as back-translation. It contains 5000 "*complex*" question-answer pairs with a minimum of two and a maximum of eight unique paraphrased responses for each question.
- VANiLLa [25] is a QA dataset that offers answers in natural language sentences. The answer sentences in this dataset are syntactically and semantically closer to the question than the triple fact. The dataset consists of over $100k$ "*simple*" questions.

Dataset	**Train**	**Test**	**Question len.**	**Answer len.**	**Vocabulary**
VQuAnDa	4000	1000	12.27	16.95	10431
ParaQA	12637	3177	12.27	17.06	12755
VANiLLa	85729	21433	8.96	8.98	50505

Table 4.1: Dataset statistics, including the (average) number of tokens per question sentence, the (average) number of tokens per answer sentence and the vocabulary list size.

Model Configurations

For simplicity, to represent the logical forms, we employ a similar grammar as in [15]. Our approach can be used with any other grammar or even directly with SPARQL queries. However, we believe it is better to employ semantic grammar from a state-of-the-art QA model. To properly train the similarity threshold module, we had to introduce negative logical forms for each question. We did that by corrupting the gold logical forms, either by replacing a random action or finding another "*similar*" logical form from the dataset based on the Levenshtein distance.

For hyperparameters, we employ a batch size of 256, a learning rate of 0.001 and we train for 100 epochs. For the optimization, we use the Noam optimizer proposed by [82], where authors use an Adam optimizer [166] with several warmup steps for the learning rate. In our case, the number of warmup steps is 4000. During optimization, we clip the gradients with a max norm of 5. We apply a dropout with a probability of 0.1 across our framework and use an embedding dimension of $d = 300$. All our modules operate under the same embedding dimension. We apply the GloVe word embedding model to our input tokens with a word embedding dimension of 300. For the Transformer encoder and decoder, we use similar configurations with [82], where $H = 6$ heads and $L = 2$ layers. The inner feed-forward linear layers have dimension $d_{ff} = 600$, $(2 * 300)$. Following the base Transformer parameters, we apply residual dropout [167] to the summation of the embeddings and the positional encodings in both encoder and decoder stacks with a rate of 0.1. The similarity threshold module receives an input of dimension 600 where here a linear layer is responsible for reducing it to 300, which is the framework dimension. Finally, for the cross attention module, we apply hyperparameters similar to the Transformer model. Our dimension remains of size $d = 300$ and again the number of heads is $H = 6$. However, we do not apply multiple layers here. Likewise, dropout is applied with probability 0.1. The number of model training parameters for VQuAnDa, ParaQA, and VANiLLa datasets are 12.9M, 14.9M, and 46.8M, respectively. The number varies due to different vocabularies for each dataset.

Grammar

For the logical forms, we employ a grammar that can be used to capture the entire context of the question with the minimum number of actions. We prefer not to reinvent the wheel, and therefore we adopted the grammar from existing state-of-the-art question answering systems [14, 15]. However, we do not employ all the actions from these works; Table 4.2 illustrates the complete grammar with all the defined actions that we used for all three answer verbalization datasets. As we can see, for a couple of actions, we also have their reverse occurrence (e.g. *find*, *find_reverse*). This is done to match the knowledge graph triple direction (*subject-predicate-object*). In some questions, we might have the subject or the object entity. Having both normal and reverse actions helps us identify the correct answer directly based on the model's predicted action. In Table 4.3, we illustrate how the actions can be used to annotate questions from all three datasets. For instance, for the VQuAnDa question "*Which sports are played in schools affiliated with the Harvest Christian Center?*" the gold logical form does include three different actions from our grammar (*find*, *filter_type* and *find_reverse*). Where the "*find_reverse*" is used to identify all the subject entities of the triple (*?subject*, *religiousAffiliation*, *Harvest_Christian_Center*). Another interesting example is the ParaQA question "*Name the office holder whose alma mater is Harvard-Westlake School and resting place is Alta Mesa Memorial Park?*" where the gold logical form contains the action "*intersection*" which allows us to identify

Action	**Description**
set → find(e, p)	set of objects part of the triples with subject e and predicate p
set → find_reverse(e, p)	set of subjects part of the triples with object e and predicate p
set → filter_type(set, tp)	filter the given set of entities based on the given type
boolean → is_in(entity, set)	check if the entity is part of the set
number → count(set)	count the number of elements in the set
set → union(set_1, set_2)	union of set_1 and set_2
set → intersection(set_1, set_2)	intersection of set_1 and set_2

Table 4.2: Predefined grammar with respective actions to generate logical forms.

Dataset	**Question**	**Logical Form**
VQuAnDa	In which team did Dave Bing and Ron Reed started their basketball career?	union(find(Dave_Bing, draftteam), find(Ron_Reed, draftteam))
	Does the white river flow into the connecticut river?	is_in(find(Connecticut_River, rightTributary), White_River_(Vermont))
	Which sports are played in schools affiliated with the Harvest Christian Center?	find(filter_type(find_reverse(Harvest_Christian_Center, religiousAffiliation), School), sport)
ParaQA	How many people have been part of Chicago Bulls team?	count(find_reverse(Chicago_Bulls, team))
	Name the office holder whose alma mater is Harvard-Westlake School and resting place is Alta Mesa Memorial Park?	intersection(filter_type(find_reverse(Harvard-Westlake_School, almamater), officeholder), find_reverse(Alta_Mesa_Memorial_Park, restingplace))
	Name all the broadcast area of the TV stations which has Rodrigues as one of the broadcast area?	find(filter_type(find_reverse(Rodrigues, broadcastArea), TelevisionStation), broadcastArea)
VANiLLa	What is the nonprofit organization where Bruce Bochy was educated?	filter_type(find(Q586449, P69), Q163740)
	Which language can Paolo Brera understand?	find(Q2423068, P1412)
	Which administrative territory is Shaun Cunnington an inhabitant of ?	filter_type(find(Q7490823, P27))

Table 4.3: Dataset examples annotated with gold logical forms.

the intersection between two sets of entities. Similarly, we also use the "*union*" action. Next, for the quantitative questions, we employ the action "*count*" which returns the length of the entity set. Finally, for verification questions, we have the "*is_in*" action, which checks whether an entity exists in a set.

Model for Comparison

We compare our framework with four baselines that have been evaluated on the considered datasets. All baselines consist of sequence to sequence architectures, a family of machine learning approaches used for language processing and often used for natural language generation tasks. The first model consists of an RNN [71] based architecture, the second uses a convolutional network [140], the third employs a Transformer network [82], while the last one is a BERT [92] model. For a fair comparison with our framework, we report the baselines' results using the question and the logical form as separate inputs considering that baselines are limited to accept both inputs together.

Evaluation Metrics

We use the same metrics as employed by the authors of the three existing datasets [24–26] on the previously mentioned baselines. The BLEU score, as defined by [141], analyzes the co-occurrences of n-grams in the reference and the proposed responses. It computes the n-gram precision for the whole dataset, which is then multiplied by a brevity penalty to penalize short translations. We report results for BLEU-4. The METEOR score introduced by [142] is based on the harmonic mean of uni-gram precision and recall, with recall weighted higher than precision. Both metrics can be in the range of 0.0 and 100, with 100 being the best score.

4.7 Results

Table 4.4 summarizes the results comparing the VOGUE framework to the previous baselines for answer verbalization. VOGUE significantly outperforms the earlier baselines for both the BLEU and METEOR scores. While for the other baselines, we perform experiments with two different inputs (Question, gold Logical Form), VOGUE is the only one that directly uses both inputs (Hybrid). As we can see, for both datasets VQuAnDa and ParaQA, all baselines perform slightly worse when receiving the question as input compared to the gold logical form. This is due to the constant input pattern templates that the logical forms have. However, this does not apply to the VANiLLa dataset since it only contains simple questions. VOGUE achieves a BLEU score of 28.76 on VQuAnDa, which is 2

	BLEU			**METEOR**		
Models	**VQuAnDa**	**ParaQA**	**VANiLLa**	**VQuAnDa**	**ParaQA**	**VANiLLa**
RNN [71] (**Q**)	15.43	22.45	16.66	53.15	58.41	58.67
RNN [71] (**LF**)	20.19	26.36	16.45	57.06	61.87	55.34
Convolutional [140] (**Q**)	21.32	25.94	15.42	57.54	60.82	61.14
Convolutional [140] (**LF**)	26.02	31.89	16.89	64.30	65.85	58.72
Transformer [82] (**Q**)	18.37	23.61	30.80	56.83	59.63	62.16
Transformer [82] (**LF**)	23.18	28.01	28.12	60.17	63.75	59.01
BERT [92] (**Q**)	22.78	26.12	31.32	59.28	62.59	62.96
BERT [92] (**LF**)	26.48	30.31	30.11	65.92	65.92	59.27
VOGUE (Ours) (**H**)	**28.76**	**32.05**	**35.46**	**67.21**	**68.85**	**65.04**

Table 4.4: Results on answer verbalization. VOGUE outperforms all existing baselines and achieves the new state of the art for both the BLEU and METEOR scores. The baseline experiment results are reported with two inputs: Question (**Q**) and gold Logical Form (**LF**), while VOGUE employs a Hybrid (**H**) approach.

points higher than the second-best BERT (LF). The same applies to the METEOR score. Regarding ParaQA, VOGUE performs slightly better than the second Convolutional (LF) on BLEU score, while on METEOR score, the margin increases to 3 points. Finally, for the VANiLLa dataset, VOGUE performs considerably better compared to other baselines.

4.8 Ablation Study

We perform several ablation studies to understand and validate the performance of our approach.

	BLEU			METEOR		
Models	**VQuAnDa**	**ParaQA**	**VANiLLa**	**VQuAnDa**	**ParaQA**	**VANiLLa**
RNN [71]	15.43	22.45	16.66	53.15	58.41	58.67
Convolutional [140]	21.32	25.94	15.42	57.54	60.82	61.14
Transformer [82]	18.37	23.61	30.80	56.83	59.63	62.16
BERT [92]	22.78	26.12	31.32	59.28	62.59	62.96
VOGUE (Ours)	**25.76**	**28.42**	**33.14**	**64.61**	**67.52**	**63.69**

Table 4.5: Results of the answer verbalization with a semantic parsing QA system. VOGUE still outperforms all baselines. For the baselines we employ only the question as input, while our framework employs the similarity threshold module to determine whether a hybrid verbalization can be performed.

Integration with Semantic Parsing based QA system

The logical forms used for the results in Table 4.4 are the gold ones, and therefore the performance of all baselines, including our framework, is boosted. In our first ablation study, we want to perform experiments in an end-to-end manner with a semantic parsing QA system, alongside the models, to understand our framework's superior performance. In this experiment, we train a simple, sequence to sequence-based semantic parser system to generate the logical forms by using the questions. As expected, the generated logical forms are not all correct, and therefore this affects the verbalization results. However, in Table 4.5, we can see that VOGUE still outperforms all baselines in this setting. An important role here plays the similarity threshold module, enabling a hybrid approach even in a real-world scenario. We can only use the question as input for the baselines since we do not have the gold logical forms. Here, it is also interesting that in two of the datasets, our framework outperforms the baselines with a more significant margin than before (cf. Table 4.5, METEOR-VQuAnDa, METEOR-ParaQA). Finally, Figure 4.3 illustrates the perplexity results, which show how well a probability distribution predicts a sample. A low perplexity indicates the probability distribution is good at predicting the sample. As we can see, our framework achieves the lowest perplexity values on all three datasets compared to other the baselines.

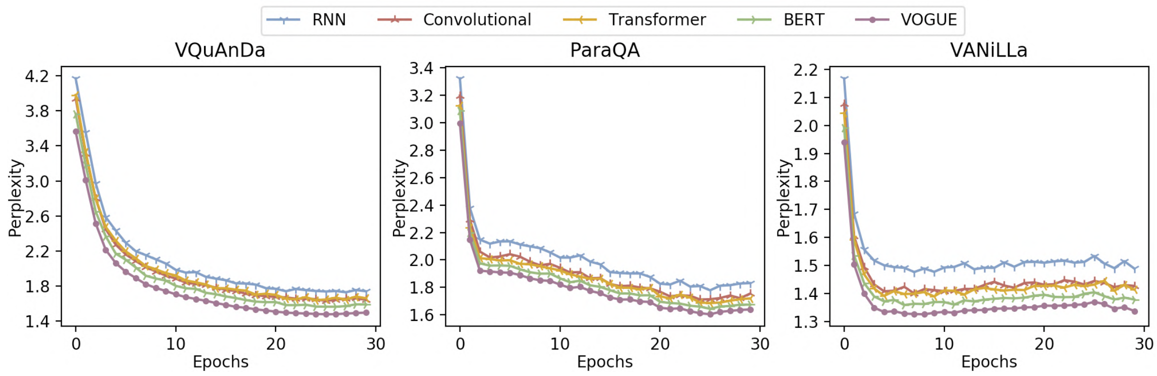

Figure 4.3: Perplexity curves for all three answer verbalization datasets.

Ablation	BLEU VQuAnDa	BLEU ParaQA	BLEU VANiLLa	METEOR VQuAnDa	METEOR ParaQA	METEOR VANiLLa
Ours	**28.76**	**32.05**	**35.46**	**67.21**	**68.85**	**65.04**
w/o Cross Attention	26.24	30.59	30.94	64.93	66.16	62.12
w/o Multi-Task Learning	25.74	28.15	29.07	62.31	63.84	61.49

Table 4.6: Ablation study results that indicate the effectiveness of cross attention and multi-task learning. The first row contains the results of the VOGUE framework when training all four modules with multi-task learning. The second and third rows selectively remove the cross attention and the multi-task learning from VOGUE. Best values in bold.

Impact of Cross Attention and Multi-Task Learning

Our second ablation experiment demonstrates the vitality of the cross-attention module and multi-task learning strategy. We first remove the cross-attention module from our framework. Instead, we only concatenate the question and logical form to generate the verbalization. As observed in Table 4.6, we obtain worse results compared to the original configuration of VOGUE. A simple concatenation does not interchange any information between the question and the logical form, and therefore the results are expected to be lower. The cross-attention module is intentionally built to determine relevance between inputs by jointly modeling the relationship between the question words and logical form actions. Next, we train all modules independently and join them on inference to understand the multi-task learning efficacy. As observed, our results have a negative impact when a multi-task learning strategy is not employed.

Similarity Threshold Task Analysis

Table 4.7 illustrates the performance of similarity threshold module. We observe that the module performs fairly well on VQuAnDa and ParaQA with F1-scores of 64.73 and 58.55, respectively. Both datasets contain complex questions. Hence, predicting the similarity between the question and the logical form is not easy. However, as long as the module's score is beyond 50, we are confident that using the similarity threshold module can improve our frameworks' answer verbalization results. For

	F1-Score		
Module	**VQuAnDa**	**ParaQA**	**VANiLLa**
Similarity Threshold	64.73	58.55	98.76

Table 4.7: Similarity threshold f1-score results for each dataset.

the VANiLLa dataset, the performance is incredibly high, with a score of 98.76. This is because the dataset contains only simple questions. Consequently, a single template pattern is employed for this dataset, and the module here has to predict if the logical form contains the correct triple relation. The task is much easier to perform compared to complex questions. Overall, the module results are pretty accurate and encourage us to apply them in our task.

4.9 Error Analysis

For the error analysis, we randomly sampled 100 incorrect predictions for human evaluation. We detail the reasons for two types of errors observed in the analysis:

Words Mischoose. A common error of VOGUE is mischoosing a word in the answer verbalization sentence. For instance, for the question "*Give me a count of everything owned by the network whose sister name is The CW?*" our framework generated the answer "*There are 156 television shows whose network's sister station is The CW.*". However, the gold reference here is "*There are 156 things whose network's sister name is The CW.*" As we can see, our framework misselected words in two parts of the sentence. The first one is the word "*things*", where it predicted "*television shows*". The second one is the word "*name*", where our model predicted "*station*". Both model predictions ("*television shows*", "*station*") are correlated, since they belong with the same context. Such errors do not heavily penalize the overall performance. For the example mentioned above, the BLEU and METEOR scores are positive, with 35.74 and 81.52, respectively.

Factual Errors. Another type of error observed is when VOGUE misses the semantic meaning and produces irrelevant results. It contributes to a major chunk of overall errors. There are two cases that can cause observed errors. The first one is the lack of reasoning for similar context data. When facing examples with the limited context in the dataset, the model would most definitely fail to reproduce the same context in the answer sentence. One can solve the issue by enriching the training data with other examples containing similar contexts. The second reason for having factual errors is when similarity threshold module fails to determine the inputs' relevance. As illustrated before, using the similarity threshold allows to successfully adopt a hybrid approach in a real-world scenario (*QA + Answer Verbalization*) and exceed any previous baseline performance.

Question	How many other home stadium are there of the soccer club whose home stadium is Luzhniki Stadium?
Logical Form	count(find_reverse(find_reverse(Luzhniki_Stadium, homeStadium), homeStadium))
Reference	There are 9 home stadiums of the soccer club whose home stadium is Luzhniki Stadium.
VOGUE	There are 9 home stadiums of the soccer club whose home stadium is Luzhniki Stadium.
Question	Who is the scientist whose academic advisor was Karl Ewald Hasse?
Logical Form	filter_type(find_reverse(Karl_Ewald_Hasse, academicAdvisor), Scientist)
Reference	The scientist whose **academic advisor** is Karl Ewald Hasse is Robert Koch.
VOGUE	The scientist whose doctoral advisor is Karl Ewald Hasse is Robert Koch.
Question	What are the movies with Daniel Waters as screenwriter?
Logical Form	filter_type(find_reverse(Daniel_Waters, screenplay), Film)
Reference	The **films** with the **screenplay written** by Daniel Waters are Batman Returns, Demolition Man (film), Hudson Hawk, The Adventures of Ford Fairlane.
VOGUE	The movies whose director is Daniel Waters are Batman Returns, Demolition Man (film), Hudson Hawk, The Adventures of Ford Fairlane.
Question	Which person designed the cars which has been designed by ASC Creative Services?
Logical Form	find(filter_type(find_reverse(ASC_Creative_Services, designCompany), Automobile), designer)
Reference	The **designers** of the **cars** whose designer company is ASC Creative Services are Warren, Michigan, Michigan, ASC Creative Services.
VOGUE	The things which have been designed by ASC Creative Services are Warren, Michigan, Michigan, ASC Creative Services.

Table 4.8: Sample output of our framework.

4.10 Case Study

We further manually inspect our framework VOGUE outputs for conducting a case study to understand the performance better. As shown in Table 4.8, we find that in the first example, VOGUE can produce the exact verbalization with the reference. Here the logical form contains three actions (*count* and two *find_reverse*), and is not a simple question. Such results indicate the superb performance of our framework. Next, we can see the question "*Who is the scientist whose academic advisor was Karl Ewald Hasse?*" here, VOGUE manages to generate almost the exact verbalization. In particular, it only mischoses a single word, which is still relevant to the context, and the generated result could be easily considered correct. Our framework here generated the answer "*The scientist whose doctoral advisor is Karl Ewald Hasse is Robert Koch.*" and the word it missed was "*academic*" where it replaced it with "*doctoral*". The following example illustrates the robustness of our framework. Here the question is "*What are the movies with Daniel Waters as screenwriter?*" and our model produces a flawless verbalization and it only complicates the words "*screenwriter*" and "*director*". It also replaces the word "*films*" with "*movies*" which can be considered synonyms and make no significant difference in verbalization. The generated response here is grammatically sound and adequately represent all the information in the question and logical form. Finally, in the last example, we can see that VOGUE has generalized in the answer verbalization compared to the reference. More precisely, the question here refers to cars designed by a company, and the reference also mentions it. However, our model produces a more general but at the same time fluent verbalization that refers to "*things*" instead of "*cars*", which again can be considered a valid response.

4.11 Summary

In this chapter, we studied the impact of jointly utilizing the question and logical form on the answer verbalization task. We empirically observed that the proposed "hybrid" approach implemented in the VOGUE framework provides a flexibility to be deployed in a real world scenario where a QA system not always will produce the correct logical form. We systematically studied the impact of our choices in the proposed architecture. For instance, the ablation study demonstrates the effectiveness of multi-task learning (for jointly training similarly threshold and answer verbalization) and cross-attention module.

CHAPTER 5

Conversational Question Answering via Multi-Task Learning

The previous two chapters (Chapter 3 & 4) contributed to the answer verbalization task, which is vital for providing more fluent answers and engaging the user in a conversation with the system. In particular, we advanced the state-of-the-art in terms of resources and approaches. We presented a dataset with multiple verbalized responses that offer flexibility to the learning process of machine learning models. We further developed a framework that leverages logical forms for improving performance. Additionally, we integrated the proposed approach with a basic QA semantic parsing system to illustrate its superior performance. Now we switch the focus to multi-turn QA systems, which contain challenging conversational scenarios. One of the primary methodologies for handling KGQA is semantic parsing, where a grammar of actions is required to build executable queries over the KG. The number of actions depends on the complexity and type of questions. While for single-turn KGQA, there have been proposed various grammars, we cannot directly apply them to the ConvQA task since they do not address conversational scenarios (e.g., clarification, co-reference, ellipsis). Moreover, it is an essential yet open challenge to effectively identify and incorporate relevant KG information such as entities, relations, and types for the ConvQA task.

In this chapter, we target the problem of conversational (complex) question answering over a large-scale knowledge graph. We explore existing related work and propose novel approaches for semantic parsing by employing state-of-the-art deep neural architectures. We divide the task into several sub-tasks and incorporate knowledge graph information. All the sub-tasks are trained via a multi-task learning paradigm.

We address the following research question in this chapter:

RQ3: How can we develop better and more efficient multi-task semantic parsing approaches for conversational question answering?

Contributions of this chapter are summarized as follows:

- We introduce two multi-task learning frameworks for conversational question answering over large scale knowledge graph.
- We propose an architecture of stacked pointer networks for incorporating knowledge graph information on conversational question answering task.

- We employ a Transformer model supplemented with a Graph Attention Network to exploit the correlations between (entity) types and predicates due to its message-passing ability between the nodes.
- We propose a novel entity recognition module that detects, links, filters, and permutes all relevant entities.
- We propose a reusable grammar for neural semantic parsing to define various logical forms that can be executed on a KG for fetching answers to conversational questions.
- We empirically study the proposed architectural design choices through an extensive evaluation, ablation study, and multiple analyses.

This chapter is based on the following publications [14, 15]:

- Joan Plepi, **Endri Kacupaj**, Kuldeep Singh, Harsh Thakkar, and Jens Lehmann. "Context Transformer with Stacked Pointer Networks for Conversational Question Answering over Knowledge Graphs." In European Semantic Web Conference, pp. 356-371. Springer, Cham, 2021. DOI: 10.1007/978-3-030-77385-4_21
- **Endri Kacupaj**, Joan Plepi, Kuldeep Singh, Harsh Thakkar, Jens Lehmann, and Maria Maleshkova. "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, pp. 850-862. 2021. DOI: 10.18653/v1/2021.eacl-main.72

The structure of the chapter is as follows: Section 5.1 introduces the work. Section 5.2 presents the related work. Section 5.3 describes the first approach, while Section 5.4 presents the second approach. We summarize in Section 5.5.

5.1 Introduction

Recently, there has been an increased demand for chatbots and voice assistants to communicate and assist humans in different domains such as chitchat, medical, news, enterprise, etc ([168]). Question answering is a common phenomenon in chatbot conversations to seek specific information. While such questions inherit a conversational context, humans also tend to ask questions that require complex reasoning to answer in a real-world scenario. The complexity of questions may differ at various granularity (e.g., simple, logical, quantitative, and comparative). Figure 5.1 presents a few examples from a complex question answering dataset with a conversational context [117]. The example dialogue has several question types and challenges. For instance, in the first turn, the user asks a simple direct question, and in the following turn, addresses a question that refers to the context from the previous interaction. Furthermore, in the last turn, the question requires ellipsis resolution to offer a multitude of complexity. Given that these questions are from the general knowledge domain, the answer can be extracted from publicly available large-scale Knowledge Graphs (KGs) such as DBpedia [1], Freebase [169], and Wikidata [2].

Neural semantic parsing approaches for question answering over KGs have been widely studied in the research community [12, 170–172]. In a given question, these approaches use a semantic parsing

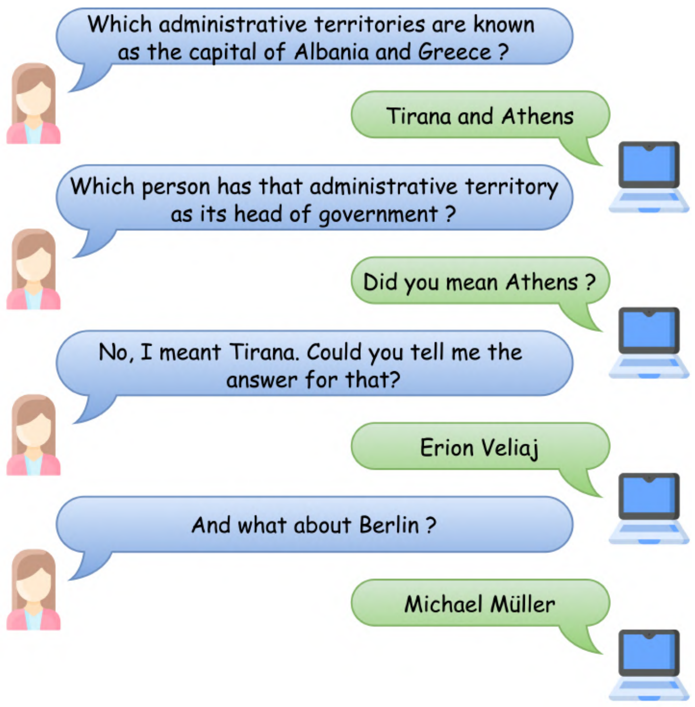
Which administrative territories are known as the capital of Albania and Greece ?
Tirana and Athens
Which person has that administrative territory as its head of government ?
Did you mean Athens ?
No, I meant Tirana. Could you tell me the answer for that?
Erion Veliaj
And what about Berlin ?
Michael Müller

Figure 5.1: Conversational Question Answering task with examples similar to CSQA dataset [117].

model to produce a logical form which is then executed on the KG to retrieve an answer. While traditional methods tend to work on small KGs [173], more recent approaches also work well on large-scale KGs [12, 13]. Often, researchers targeting large scale KGs focus on a stepwise method by first performing entity linking and then train a model to learn the corresponding logical form for each question type [12, 174].

The state-of-the-art for semantic parsing approaches decomposes the semantic parsing process into two stages [13]. First, a logical form is generated based on low-level features and then the missing details are filled by considering both the question and the template. Other approaches [12, 171, 174] first employ an entity linking model to identify entities in the question and subsequently use another model to map the question to a logical form. Recent works [13, 175] point out that the modular approaches suffer from the common issue of error propagation along the QA pipeline, resulting in accumulated errors. Work in [13] argues that the stepwise approaches have two significant issues. First, errors in upstream sub-tasks (e.g., entity detection and linking, relation classification) are propagated to downstream ones (e.g., semantic parsing), resulting in accumulated errors. For example, case studies in previous works [12, 174, 176] show that entity linking error is one of the significant errors leading to the wrong results in the question-answering task. Second, when models for the sub-tasks are learned independently, the supervision signals cannot be shared among the models for mutual benefits. To mitigate the limitations of the stepwise approach, work in [13] proposed a multi-task learning framework where a pointer-equipped semantic parsing model is designed to resolve co-reference in conversations, and intuitively, empower joint learning with a type-aware entity detection model. The ConvQA approaches we propose in this chapter handle various limitations of existing works and introduce solutions that advance the state-of-the-art.

5.2 Related Work

We point to the survey by Gao et al. [177] that provides a holistic overview of neural approaches in conversational AI. In this chapter, we stick to our closely related work, i.e., semantic parsing and multi-task learning approaches in conversations. We also briefly refer to other approaches.

Semantic Parsing and Multi-Task Learning Approaches

Our work lies in the areas of semantic parsing and neural approaches for question answering over KGs. Works in [58, 63, 175, 178] use neural approaches to solve the task of QA.

Lukovnikov et al. [178] introduces an approach that splits the question into spans of tokens to match the tokens to their respective entities and predicates in the KG. The authors merge the word and character-level representation to discover better matching in entities and predicates. Candidate subjects are generated based on n-grams matching with words in the question, and then pruned based on predicted predicates. However, their experiments are focused on simple questions.

Zhang et al. [175] propose a probabilistic framework for QA systems and experiment on a new benchmark dataset. The framework consists of two modules. The first one model the probability of the topic entity y, constrained on the question. The second module reasons over the KG to find the answer a, given the topic entity y, which is found in the first step and question q. Graph embeddings are used to model the subgraph related to the question and calculate the answer's distribution depending on the question q and the topic y.

Liang et al. [171] introduce neural symbolic machine (NSM), which contains a neural sequence to sequence network referred also as the "programmer", and a symbolic non-differentiable LISP interpreter ("computer"). The model is extended with a key-value memory network, where keys and values are the output of the sequence model in different encoding or decoding steps. The NSM model is trained using the REINFORCE algorithm with weak supervision and evaluated on the WebQuestionsSP dataset [126].

Saha et al. [117] propose a hybrid model of the HRED model [179] and the key-value memory network model [180]. The model consists of three components. The first one is the Hierarchical Encoder, which computes a representation for each utterance. The following module is a higher-level encoder that computes a representation for the context. The second component is the Key-Value Memory Network which stores each candidate tuples as a key-value pair. The key contains the concatenated embedding of the relation and the subject. In contrast, the value includes the embedding of the object. The last component is the decoder used to create an end-to-end solution and produce various answers.

Guo et al. [12] also present an approach that maps utterances to logical forms. Their model consists of a sequence to sequence network, where the encoder produces the embedding of the utterance, and the decoder generates the sequence of actions. Authors introduce a dialogue memory management to handle the entities, predicates, and logical forms that are referred from a previous interaction. The dialogue memory aims to solve the problem of interaction history when an entity, predicate, or type is needed to be instantiated in a specific step of the action sequence.

Shen et al. [13] proposed a multi-task learning framework that learns type-aware entity detection and pointer-equipped logical form generation simultaneously. The multi-task learning framework takes advantage of the supervision from the sub-tasks.

Marion et al. [181] addressed the problem of weakly-supervised ConvQA over KGs and proposed an object aware Transformer model that can receive structured input (i.e., JSON format) that allows it to process KG contextual information. They also present a KG grammar for semantic parsing QA.

Thirukovalluru et al. [182] present a neural semantic parsing decoder that employs additional KG information for ConvQA. The authors propose a knowledge injection layer that combines KG embeddings into the state at each decoding step. For informing the decoder regarding the expected structure of the KG, they include an attention layer on random, k-hops knowledge walks from encountered entities at each decoding step.

Other Approaches

There has been extensive research for task-oriented dialog systems such as [154] that induces joint text and knowledge graph embeddings to improve task-oriented dialogues in the domains such as restaurant and flight booking. Work present in [16] proposes another dataset, "ConvQuestions" for conversations over KGs along with an unsupervised model. Some other datasets include CANARD and TREC CAsT [183]. Overall, several approaches are proposed for conversational QA, and in this work, we closely stick to semantic parsing and multi-task learning for our approaches comparison and contributions.

5.3 Context Transformer with Stacked Pointer Networks

The first approach we present is CARTON (**C**ontext tr**A**nsforme**R** s**T**acked p**O**inter **N**etworks), a multi-task learning framework consisting of a context Transformer model extended with a stack of pointer networks for multi-task neural semantic parsing. Our framework handles semantic parsing using the context Transformer model while the stacked pointer networks handle the remaining tasks such as type prediction, predicate prediction, and entity detection. Our proposed framework is inspired by [13]; however, we differ considerably on the following points: 1) CARTON's stacked pointer networks incorporate knowledge graph information for performing reasoning and do not rely only on the conversational context as MaSP does. 2) The stacked pointer network architecture is used intentionally to provide the flexibility for handling out-of-vocabulary [184] entities, predicates, and types that are unseen during training. Our ablation study 5.3.5 further supports our choices. The MaSP model does not cover out-of-vocabulary knowledge since the model was not intended to have this flexibility. 3) CARTON's supervision signals are propagated in sequential order, and all the components use the signal forwarded from the previous component. 4) We employ semantic grammar with new actions for generating logical forms. While [13] operates nearly with the same grammar as [12].

CARTON achieves new state of the art results on eight out of ten question types from a large-scale conversational question answering dataset. We evaluate CARTON on the Complex Sequential Question Answering (CSQA) [117] dataset consisting of conversations over linked QA pairs. The dataset contains 200K dialogues with 1.6M turns, and over 12.8M entities. Our implementation, the annotated dataset with proposed grammar, and results are on a public github[1].

The structure of the section is as follows: Subsection 5.3.1 presents the proposed framework. Subsection 5.3.2 describes the multi-task learning process. Subsection 5.3.3 describes the experimental setup, while Subsection 5.3.4 the experiments results. Subsection 5.3.5 provides a detailed ablation study. An error analysis is on Subsection 5.3.6. We provide a brief summary in Subsection 5.3.7.

[1] https://github.com/endrikacupaj/CARTON

5.3.1 Approach

Our focus is on conversations containing complex questions that can be answered by reasoning over a large-scale KG. The training data consists of utterances u and the answer label a. We propose a semantic parsing approach, where the goal is to map the utterance u into a logical form z, depending on the conversation context. A stack of three pointer networks is used to fill information extracted from the KG. The final generated logical form aims to fetch the correct answer once executed on the KG. Figure 5.2 illustrates the overall architecture of CARTON framework.

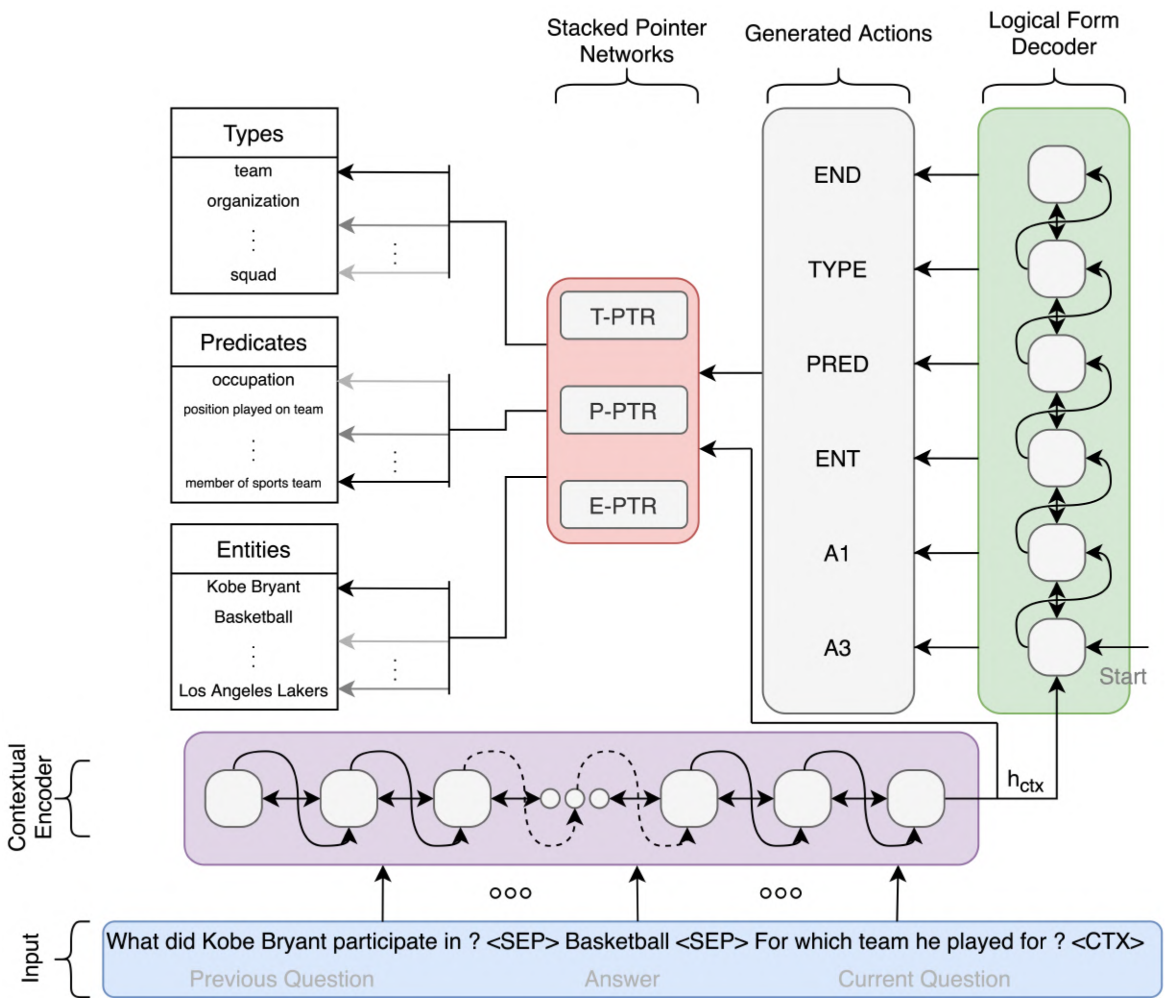

Figure 5.2: CARTON (**C**ontext tr**A**nsforme**R** s**T**acked p**O**inter **N**etworks) architecture. It consists of three modules: 1) A Transformer-based contextual encoder finds the representation of the current context of the dialogue. 2) A logical form decoder generates the pattern of the logical forms defined in Table 5.1. 3) The stacked pointer network initializes the KG items to fetch the correct answer.

Grammar

We propose a grammar with various actions as shown in Table 5.1 which can result in different logical forms that can be executed on the KG. Our grammar definition is inspired by [12] which MaSP [13] also employs. However, we differ in many semantics of the actions and we even defined completely new actions. For example, *find* action is split into *find(e, p)* that corresponds to finding an edge with predicate *p* and subject *e*; and *find_reverse(e, p)* finds an edge with predicate *p* and object *e*. Moreover, *per_type* is not defined by [12] in their grammar. Table 5.2 indicates some (complex) examples from CSQA dataset [117] with gold logical form annotations using our predefined grammar. Following [185], each action definition is represented by a function that is executed on the KG, a list of input parameters, and a semantic category that corresponds to the output of the function. For example, *set* → *find(*e, p*)*, it has a *set* as a semantic category, and a function *find* with input parameters *e*, *p*. We believe that the defined actions are sufficient for creating sequences that cover complex questions and we provide empirical evidences in Section 5.3.4. Every action sequence can be parsed into a tree, where the model recursively writes the leftmost non-terminal node until the whole tree is complete. The same approach is followed to execute the action sequence, except that the starting point is the tree leaves.

Action	**Description**
set → find(e, p)	set of objects (entities) with subject e and predicate p
set → find_reverse(e, p)	set of subjects (entities) with object e and predicate p
set → filter_by_type(set, tp)	filter the given set of entities based on the given type
set → filter_mult_types(set_1, set_2)	filter the given set of entities based on the given set of types
boolean → is_in(set, entity)	check if the entity is part of the set
boolean → is_subset(set_1, set_2)	check if set_2 is subset of set_1
number → count(set)	count the number of elements in the set
dict → per_type(p, tp_1, tp_2)	extracts a dictionary, where keys are entities of $type_1$ and values are the number of objects of $type_2$ related with p
dict → per_type_rev(p, tp_1, tp_2)	extracts a dictionary, where keys are entities of $type_1$ and values are the number of subjects of $type_2$ related with p
set → greater(num, dict)	set of entities that have greater count than *num*
set → lesser(num, dict)	set of entities that have lesser count than *num*
set → equal(num, dict)	set of entities that have equal count with *num*
set → approx(num, dict)	set of entities that have approximately same count with *num*
set → argmin(dict)	set of entities that have the most count
set → argmax(dict)	set of entities that have the least count
set → union(set_1, set_2)	union of set_1 and set_2
set → intersection(set_1, set_2)	intersection of set_1 and set_2
set → difference(set_1, set_2)	difference of set_1 and set_2

Table 5.1: Predefined grammar with respective actions to generate logical forms.

Question Type	**Question**	**Logical Forms**
Simple Question (Direct)	Q1: Which administrative territory is the birthplace of Antonio Reguero ?	filter_type(find(Antonio Reguero, place of birth), administrative territorial entity)
Simple Question (Ellipsis)	Q1: Which administrative territories are twin towns of Madrid ? A1: Prague, Moscow, Budapest Q2: And what about Urban Community of Brest?	filter_type(find(Urban Community of Brest, twinned administrative body), administrative territorial entity)
Simple Question (Coreferenced)	Q1: What was the sport that Marie Pyko was a part of ? A1: Association football Q2: Which political territory does that person belong to ?	filter_type(find(Marie Pyko, country of citizenship), political territorial entity)
Quantitative Reasoning (Count) (All)	Q1: How many beauty contests and business enterprises are located at that city ? A1: Did you mean Caracas? Q2: Yes	count(union(filter_type(find_reverse(Caracas, located in), beauty contest), filter_type(find_reverse(Caracas, located in), business enterprises)))
Quantitative Reasoning (All)	Q1; Which political territories are known to have diplomatic connections with max number of political territories ?	argmax(per_type(diplomatic relation, political territorial entity, political territorial entity))
Comparative Reasoning (Count) (All)	Q1: How many alphabets are used as the scripts for more number of languages than Jawi alphabet ?	count(greater(count(filter_type(find(Jawi alphabet, writing system), language)), per_type(writing system, alphabet, language)))
Comparative Reasoning (All)	Q1: Which occupations were more number of publications and works mainly about than composer ?	greater(union(per_type(main subject, occupation, publication), per_type(main subject, occupation, work)), count(filter_multi_types(find_reverse(composer, main subject), {publication, work})))
Verification	Q1: Was Geir Rasmussen born at that administrative territory ?	is_in(find(Geir Rasmussen, place of birth), Chicago)

Table 5.2: Examples from the CSQA dataset [117], annotated with gold logical forms.

Context Transformer

Here we describe the semantic parsing part of CARTON, which is a context Transformer. The Transformer receives input a conversation turn that contains the context of the interaction and generates a sequence of actions. Formally, an interaction I consists of the question q that is a sequence $x = \{x_1, \ldots, x_n\}$, and a label l that is a sequence $y = \{y_1, \ldots, y_m\}$. The network aims to model the conditional probability $p(y|x)$.

Contextual Encoder. In order to cope with co-reference and ellipsis phenomena, we require to include the context from the previous interaction in the conversation turn. To accomplish that, the input to the contextual encoder is the concatenation of three utterances from the dialog turn: 1) the previous question, 2) the previous answer, and 3) the current question. Every utterance is separated from one another using a $< SEP >$ token. A special context token $< CTX >$ is appended at the end where the embedding of this utterance is used as the semantic representation for the entire input question. Given an utterance q containing n words $\{w_1, \ldots, w_n\}$, we use GloVe [162] to embed the words into a vector representation space of dimension d_{emb}. More specifically, we get a sequence $x = \{x_1, \ldots, x_n\}$ where x_i is given by,

$$x_i = GloVe(w_i), \tag{5.1}$$

and $x_i \in \mathbb{R}^{d_{emb}}$. Next, the word embeddings x, are forwarded as input to the contextual encoder, that uses the multi-head attention mechanism from the Transformer network [82]. The encoder outputs the contextual embeddings $h = \{h_1, \ldots, h_n\}$, where $h_i \in \mathbb{R}^{d_{emb}}$, and it can be written as:

$$h = encoder(x; \theta^{(enc)}), \tag{5.2}$$

where $\theta^{(enc)}$ are the trainable parameters of the contextual encoder.

Logical Form Decoder. For the decoder, we likewise utilize the Transformer architecture with a multi-head attention mechanism. The decoder output is dependent on contextual embeddings h originated from the encoder. The decoder detects each action and general semantic category from the KG, i.e., the decoder predicts the correct logical form, without specifying the entity, predicate, or type. Here, the decoder vocabulary consists of $V = \{A_0, A_1, \ldots, A_{18}, entity, predicate, type\}$ where $A_0, A_1, \ldots, A_{18}$ are the short names of actions in Table 5.1. The goal is to produce a correct logical form sequence. The decoder stack is supported by a linear and a softmax layer to estimate the probability scores, i.e., we can define it as:

$$\begin{aligned} s^{(dec)} &= decoder(h; \theta^{(dec)}), \\ p_t &= softmax(\mathbf{W}^{(dec)} s_t^{(dec)}), \end{aligned} \tag{5.3}$$

where $s_t^{(dec)}$ is the hidden state of the decoder in time step t, $\theta^{(dec)}$ are the model parameters, $\mathbf{W}^{(dec)} \in \mathbb{R}^{|V| \times d_{emb}}$ are the weights of the feed-forward linear layer, and $p_t \in \mathbb{R}^{|V|}$ is the probability distribution over the decoder vocabulary for the output token in time step t.

Stacked Pointer Networks

As we mentioned, the decoder only outputs the actions without specifying any KG items. To complete the logical form with instantiated semantic categories, we extend our model with an architecture of stacked pointer networks [184]. The architecture consists of three-pointer networks and each one of them is responsible for covering one of the major semantic categories (types, predicates, and entities) required for completing the final executable logical form against the KG.

The first two pointer networks of the stack are used for predicates and types semantic category and follow a similar approach. The vocabulary and the inputs are the entire predicates and types of the KG. We define the vocabularies, $V^{(pd)} = \{r_1, \dots, r_{n_{pd}}\}$ and $V^{(tp)} = \{\tau_1, \dots, \tau_{n_{tp}}\}$, where n_{pd} and n_{tp} is the total number of predicates and types in the KG, respectively. To compute the pointer scores for each predicate or type candidate, we use the current hidden state of the decoder and the context representation. We model the pointer networks with a feed-forward linear network and a softmax layer. We can define the type and predicate pointers as:

$$\begin{aligned} p_t^{(pd)} &= softmax(\boldsymbol{W}_1^{(pd)} v_t^{(pd)}), \\ p_t^{(tp)} &= softmax(\boldsymbol{W}_1^{(tp)} v_t^{(tp)}), \end{aligned} \tag{5.4}$$

where $p_t^{(pd)} \in \mathbb{R}^{|V^{(pd)}|}$ and $p_t^{(tp)} \in \mathbb{R}^{|V^{(tp)}|}$ are the probability distributions over the predicate and type vocabularies respectively. The weight matrices $\boldsymbol{W}_1^{(pd)}$, $\boldsymbol{W}_1^{(tp)} \in \mathbb{R}^{1 \times d_{kg}}$. Also, v_t is a joint representation that includes the knowledge graph embeddings, the context and the current decoder state, computed as:

$$\begin{aligned} v_t^{(pd)} &= tanh(\boldsymbol{W}_2^{(pd)} [s_t; h_{ctx}] + r), \\ v_t^{(tp)} &= tanh(\boldsymbol{W}_2^{(tp)} [s_t; h_{ctx}] + \tau), \end{aligned} \tag{5.5}$$

where the weight matrices $\boldsymbol{W}_2^{(pd)}, \boldsymbol{W}_2^{(tp)} \in \mathbb{R}^{d_{kg} \times 2d_{emb}}$, transform the concatenation of the current decoder state s_t with the context representation h_{ctx}. We denote with d_{kg} the dimension used for knowledge graph embeddings. $r \in \mathbb{R}^{d_{kg} \times |V^{(pd)}|}$ are the predicate embeddings and $\tau \in \mathbb{R}^{d_{kg} \times |V^{(tp)}|}$ are the type embeddings. $tanh$ is the non-linear layer. Please note, that the vocabulary of predicates and types can be updated during evaluation, hence the choice of pointer networks.

The third pointer network of the stack is responsible for the entity prediction task. Here we follow a slightly different approach due to the massive number of KG entities. Predicting a probability distribution over KG with a considerable number of entities is not computationally feasible. For that reason, we decrease the size of entity vocabulary during each logical form prediction. In each conversation, we predict a probability distribution **only** for the entities that are part of the context. For each conversation turn, our entity "memory" involves entities from the previous question, previous answer, and current question. The probability distribution over the entities is then calculated in the same way as for predicates and types, where the softmax is:

$$p_t^{(ent)} = softmax(\boldsymbol{W}_1^{(ent)} v_t^{(ent)}), \tag{5.6}$$

where $p_t^{(ent)} \in \mathbb{R}^{|V_k^{(ent)}|}$, and $V_k^{(ent)}$ is the set of entities for the k^{th} conversation turn. The weight matrix $\boldsymbol{W}_1^{(ent)} \in \mathbb{R}^{1 \times d_{kg}}$ and the vector v_t is then computed following the same equations as before:

$$v_t^{(ent)} = tanh(\boldsymbol{W}_2^{(ent)}([s_t; h_{ctx}]) + e_k) \tag{5.7}$$

where e_k is the sequence of entities for the k^{th} conversation turn. In general, the pointer networks are robust to handle a different vocabulary size for each time step [184]. Moreover, given the knowledge graph embeddings, our stacked pointer networks select the relevant items from the knowledge graph depending on the conversational context. In this way, we incorporate knowledge graph embeddings in order to perform any reasoning and do not rely only on utterance features. Furthermore, the [13] utilizes a single pointer network that only operates on the input utterance to select the already identified entities. Our stacked pointer networks do not use the input utterance but rather directly rely on the knowledge graph semantic categories (types, predicates, and entities).

5.3.2 Multi-Task Learning

For each time step, we have four different predicted probability distributions. The first is the decoder output over the logical form's vocabulary. The three others are from the stacked pointer networks for each semantic category (entity, predicate, and type). Finally, we define CARTON loss function as:

$$Loss_t = -\frac{1}{m}\sum_{i=1}^{m}\left(log\ p_{i\ [i=y_i^{(t)}]} + \sum_{c\in\{ent,pd,tp\}} I_{[y_i^{(t)}=c]}\ logp^{(c)}_{i\ [i=y_i^{(c_t)}]}\right), \tag{5.8}$$

where $Loss_t$ is the loss function computed for the sequence in time step t, m is the length of the logical form, $y^{(t)}$ is the gold sequence of logical form, and $y^{(c_t)}$ is the gold label for one of the semantic categories $c \in \{ent, pd, tp\}$.

5.3.3 Experimental Setup

In this subsection, we provide the setup for the experiments. We further describe the resources and metrics we employ.

Dataset and Experiment Settings

We conduct our experiments on the Complex Sequential Question Answering (CSQA) dataset[2] [117]. CSQA was built on the Wikidata KG. The CSQA dataset consists of around 200K dialogues where each partition, train, valid, and test contains 153K, 16K, 28K dialogues, respectively. The questions in the CSQA dataset involve complex reasoning on Wikidata to determine the correct answer. The different question types in the dataset are simple questions, logical reasoning, verification, quantitative reasoning, comparative reasoning, and clarification. We can have different subtypes for each one of them, such as direct, indirect, coreference, and ellipsis questions. We stick to one dataset in experiments for the following reasons: 1) all the baseline approaches have been trained and tested only on the CSQA dataset. Hence, for a fair evaluation and comparison of our approach inheriting the

[2] https://amritasaha1812.github.io/CSQA

evaluation settings same as [13, 117], we stick to the CSQA dataset. 2) other approaches [16, 183] on datasets such as ConvQuestions, and TREC CAsT, are not compatible with semantic parsing methods and, therefore, we cannot retrain existing ConvQA works [12, 13, 117] on these datasets due to their missing logical forms employed by each of these models.

We incorporate a semi-automated preprocessing step to annotate the CSQA dataset with gold logical forms. For each question type and subtype in the dataset, we create a general template with a pattern sequence that the actions should follow. Thereafter, for each question, we follow a set of rules to create the specific gold logical form that extracts the gold sequence of actions based on the type of the question. The actions used for this process are the ones in Table 5.1.

Model Configurations

For the Transformer network, we use the configurations from [82]. Our model dimension is $d_{model} = 512$, with a total number of $H = 8$ heads and layers $L = 4$. The inner feed-forward linear layers have dimension $d_{ff} = 2048$, (4×512). Following the base Transformer parameters, we apply residual dropout to the summation of the embeddings and the positional encodings in both encoder and decoder stacks with a rate of 0.1. On the other hand, the pointer networks also use a dropout layer for the linear projection of the knowledge graph embeddings. We randomly initialize the embeddings for predicates and types, and we jointly learn them during training. The KG embeddings dimension of predicate and types match the Transformer model dimension, $d_{kg} = 512$. However, for the entities, we follow a different initialization. Due to a significantly high number of the entities, learning the entity embeddings from scratch was inefficient and resulted in poor performance. Therefore, to address this issue, we initialized the entity embeddings using sentence embeddings that implicitly use underlying hidden states from BERT network [92]. For each entity, we treat the tokens that it contains as a sentence, and we feed that as an input. We receive as output the entity representation with a dimension $d_{ent} = 768$. Next, we feed this into a linear layer that learns, during training, to embed the entity into the same dimension as the predicates and types. We did not perform any hyperparameter tuning through third libraries to impact the results; hence, the performance is purely due to the underlying model's efficacy.

Models for Comparison

To compare the CARTON framework, we use the last three baselines that have been evaluated on the employed dataset. The authors of the CSQA dataset introduce the first baseline: HRED+KVmem [117] model. HRED+KVmem employs a seq2seq [78] model extended with memory networks [186, 187]. The model uses HRED model [179] to extract dialog representations and extends it with a Key-Value memory network [180] for extracting information from KG. Next, D2A [12] uses a semantic parsing approach based on a seq2seq model, extended with dialog memory manager to handle different linguistic problems in conversations such as ellipsis and co-reference. Finally, MaSP [13] is also a semantic parsing approach. It is important to note that CARTON's number of training parameters is 10.7M, compared to MaSP (vanilla) with 15M.

Evaluation Metrics

To evaluate CARTON, we use the same metrics as employed by the authors of the CSQA dataset [117] and previous baselines. We use the F1-score for questions that have an answer composed by a set of entities. Accuracy is used for the question types whose answer is a number or a boolean value (YES/NO).

Methods	**HRED-KV**	**D2A**	**MaSP**	**CARTON (ours)**	**Δ**
# Train Param	-	-	15M	10.7M	
Question Type (QT)	**F1 Score**				
Overall	9.39%	66.70%	79.26%	**81.35%**	**+2.09%**
Clarification	16.35%	35.53%	**80.79%**	47.31%	-33.48%
Comparative Reasoning (All)	2.96%	48.85%	**68.90%**	62.00%	-6.90%
Logical Reasoning (All)	8.33%	67.31%	69.04%	**80.80%**	+11.76%
Quantitative Reasoning (All)	0.96%	56.41%	73.75%	**80.62%**	+6.87%
Simple Question (Coreferenced)	7.26%	57.69%	76.47%	**87.09%**	+10.62%
Simple Question (Direct)	13.64%	78.42%	85.18%	**85.92%**	+0.74%
Simple Question (Ellipsis)	9.95%	81.14%	83.73%	**85.07%**	+1.34%
Question Type (QT)	**Accuracy**				
Overall	14.95%	37.33%	45.56%	**61.28%**	**+15.72**%
Verification (Boolean)	21.04%	45.05%	60.63%	**77.82%**	+17.19%
Quantitative Reasoning (Count)	12.13%	40.94%	43.39%	**57.04%**	+13.65%
Comparative Reasoning (Count)	8.67%	17.78%	22.26%	**38.31%**	+16.05%

Table 5.3: Comparisons among baseline models on the CSQA dataset having 200K dialogues with 1.6M turns, and over 12.8M entities.

5.3.4 Results

We report our empirical results in Table 5.3, and conclude that CARTON outperforms baselines average on all question types (row "overall" in the table). We dig deeper into the accuracy per question type to understand the overall performance. Compared to the previous state-of-the-art (MaSP), CARTON performs better on eight out of ten question types. CARTON is leading MaSP in question type categories such as *Logical Reasoning (All)*, *Quantitative Reasoning (All)*, *Simple Question (Coreferenced)*, *Simple Question (Direct)*, *Simple Question (Ellipsis)*, *Verification (Boolean)*, *Quantitative Reasoning (Count)*, and *Comparative Reasoning (Count)*. Whereas, MaSP retains the state of the art for the categories of *Clarification* and *Comparative Reasoning (All)*. The main reason for weak results in *Comparative Reasoning (All)* is that our preprocessing step finds limitation in covering this question type and is one of the shortcoming of our proposed grammar[3]. We investigated several reasonable ways to cover *Comparative Reasoning (All)* question type. However, it was challenging to produce a final answer set identical to the gold answer set. For instance, consider the question *"Which administrative territories have diplomatic relations with around the same number of administrative territories than Albania?"* that includes logic operators like "around the same number", which is ambiguous because CARTON needs to look for the correct answer in a range of the numbers. Whereas, MaSP uses a BFS method to search the gold logical forms and performance is superior to CARTON. The limitation with *Comparative Reasoning* question type also affects CARTON's performance in the *Clarification* question type where a considerable number of questions correspond to *Comparative*

[3] When we applied the preprocessing step over the test set, we could not annotate the majority of the examples for the *Comparative Reasoning (All)* question type.

Reasoning. Based on analysis, we outline the following two reasons for CARTON's outperformance over MaSP: First, the MaSP model requires to perform entity recognition and linking to generate the correct entity candidate. Even though MaSP is a multi-task model, errors at entity recognition step will still be propagated to the underlying network. CARTON is agnostic of such a scenario since the candidate entity set considered for each conversation turn is related to the entire relevant context (the previous question, answer, and current question). In CARTON, entity detection is performed only by stacked pointer networks. Hence no error propagation related to entities affects previous steps of the framework. Second, CARTON uses better supervision signals than MaSP. As mentioned earlier, CARTON supervision signals propagate in sequential order, and all components use the signal forwarded from the previous components. In contrast, the MaSP model co-trains entity detection and semantic parsing with different supervision signals.

5.3.5 Ablation Study

An ablation study is conducted to support our architectural choices of CARTON. To do so, we replace the stacked pointer networks module with simple classifiers. In particular, predicates and types are predicted using two linear classifiers and the representations from the contextual encoder. Table 5.4 illustrates that the modified setting (w/o St. Pointers) significantly under-performs compared to CARTON in all question types. The stacked pointer networks generalize better in the test set due to their ability to learn meaningful representations for the KG items and align learned representations with the conversational context. While classifiers thoroughly learn to identify common patterns between examples without incorporating any information from the KG. Furthermore, our framework's improved results are implied from the ability of stacked pointer networks to handle out-of-vocabulary entities, predicates, and types that are unseen during training.

Question Type (QT)	**CARTON**	**W/o St. Pointers**
Clarification	47.31%	42.47%
Comparative Reasoning (All)	62.00%	55.82%
Logical Reasoning (All)	80.80%	68.23%
Quantitative Reasoning (All)	80.62%	71.59%
Simple Question (Coreferenced)	87.09%	85.28%
Simple Question (Direct)	85.92%	83.64%
Simple Question (Ellipsis)	85.07%	82.11%
Verification (Boolean)	77.82%	70.38%
Quantitative Reasoning (Count)	57.04%	51.73%
Comparative Reasoning (Count)	38.31%	30.87%

Table 5.4: CARTON ablation study. "W/o St. Pointer" column shows results when stacked pointers in CARTON is replaced by classifiers.

5.3.6 Error Analysis

We now present a detailed analysis of CARTON by reporting some additional metrics. Table 5.5 reports the accuracy of predicting the KG items such as entity, predicate, or type using CARTON. The prediction accuracy of KG items is closely related to the performance of our model, as shown in Table 5.3. For example, in the *Quantitative (All)* question type (Table 5.3), predicting the correct type has an accuracy of 73.46% which is lowest compared to other question types. The type prediction is essential in such category of questions, where a typical logical form possibly is: *"argmin, find_tuple_counts, predicate, type1, type2"*. Filtering with the wrong type fetches erroneous results. Please note, there is no "entity" involved in the logical forms of *Quantitative (All)* question type. Hence, no entity accuracy is reported.

Question Type (QT)	**Entity**	**Predicate**	**Type**
Clarification	36.71%	94.76%	80.79%
Comparative Reasoning (All)	67.63%	97.92%	77.57%
Logical Reasoning (All)	64.7%	83.18%	91.56%
Quantitative Reasoning (All)	-	98.46%	73.46%
Simple Question (Coreferenced)	81.13%	91.09%	80.13%
Simple Question (Direct)	86.07%	91%	82.19%
Simple Question (Ellipsis)	98.87%	92.49%	80.31%
Verification (Boolean)	43.01%	94.72%	-
Quantitative Reasoning (Count)	79.60%	94.46%	79.51%
Comparative Reasoning (Count)	70.29%	98.05%	78.38%

Table 5.5: CARTON stacked pointer networks results for each question type. We report CARTON's accuracy in predicting the KG items such as entity, predicate, or type.

Another interesting result is the high accuracy of the entities and predicates in *Comparative (Count)* questions. Also, the accuracy of type detection is 78.38%. However, these questions' accuracy was relatively low, only 38.31%, as reported in Table 5.3. We believe that improved accuracy is mainly affected due to the mapping process of entities, predicates, and types to the logical forms that is followed to reach the correct answer. Another insightful result is on *Simple Question (Ellipsis)*, where CARTON has a high entity accuracy compared with *Simple Question (Direct/Coreferenced)*. A possible reason is the short length of the question, making it easier for the model to focus on the right entity. Some example of this question type is *"And what about Bonn?"*, where the predicate is extracted from the previous utterance of the question.

We compute the accuracy of the decoder which is used to find the correct patterns of the logical forms. We also calculate the accuracy of the logical forms after the pointer networks initialize the KG items. We report an average accuracy across question types for generating logical form (by decoder) as 97.24%, and after initializing the KG items, the average accuracy drops to 75.61%. Higher accuracy of logical form generation shows the decoder's effectiveness and how the Transformer network can extract the correct patterns given the conversational context. Furthermore, it also justifies that the higher error percentage is generated while initializing the items from the KG. When sampling some of the wrong logical forms, we found out that most of the errors were generated from initializing a similar predicate or incorrect order of the types in the logical actions.

5.3.7 Synopsis

In this work, we used a Transformer-based model to generate logical forms. The decoder was extended with a stack of pointer networks in order to include information from the large-scale KG associated with the dataset. Given the conversational context, the stacked pointer networks predict the exact KG item required in a particular position of the action sequence. We empirically demonstrate that our model performs the best in several question types and how entity and type detection accuracy affect the performance. In the next section, we describe the second proposed approach for ConvQA.

5.4 Multi-Task Semantic Parsing with Transformer and Graph Attention Networks

We present LASAGNE (mu**L**ti-task sem**A**ntic par**S**ing with tr**A**nsformer and **G**raph atte**N**tion n**E**tworks) - a multi-task learning framework consisting of a Transformer model extended with Graph Attention Networks (GATs) [107] for multi-task neural semantic parsing. Our framework handles semantic parsing using the Transformer [82] model similar to previous approaches. However, in LASAGNE we introduce the following two novel contributions: 1) the Transformer model is supplemented with a Graph Attention Network to exploit the correlations between (entity) types and predicates due to its message-passing ability between the nodes. 2) We propose a novel entity recognition module that detects, links, filters, and permutes all relevant entities. Shen et al. [13] uses a pointer equipped decoder that learns and identifies the relevant entities for the logical form using only the encoder's information. In contrast, we use both sources of information, i.e., the entity detection module and the encoder, to filter and permute the relevant entities for a logical form. This avoids re-learning entity information in the current question context and relies on the entity detection module's information. Our empirical results show that the proposed novel contributions lead to substantial performance improvements.

LASAGNE achieves state-of-the-art results in eight out of ten question types on the Complex Sequential Question Answering (CSQA) [117] dataset consisting of conversations over linked QA pairs. The dataset contains 200K dialogues with 1.6M turns, and over 12.8M entities from Wikidata[4]. Our implementation, the annotated dataset with the proposed grammar, and the results are publicly available to facilitate reproducibility and reuse[5].

The structure of the section is as follows: Subsection 5.4.1 presents the proposed framework. Subsection 5.4.2 describes the multi-task learning process. Subsection 5.4.3 describes the experimental setup, while Subsection 5.4.4 the experiments results. Subsection 5.4.5 provides a detailed ablation study. A task analysis is on Subsection 5.4.7, and an error analysis on Subsection 5.4.7. We provide a brief summary in Subsection 5.4.8.

[4] https://www.wikidata.org/
[5] https://github.com/endrikacupaj/LASAGNE

5.4.1 Approach

In a conversation, the input data consists of utterances u and their answers a, extracted from the knowledge graph. Our framework LASAGNE employs a multi-task semantic parsing approach. In particular, it maps the utterance u to a logical form z, depending on the conversation context. Figure 5.3 shows the architecture of LASAGNE.

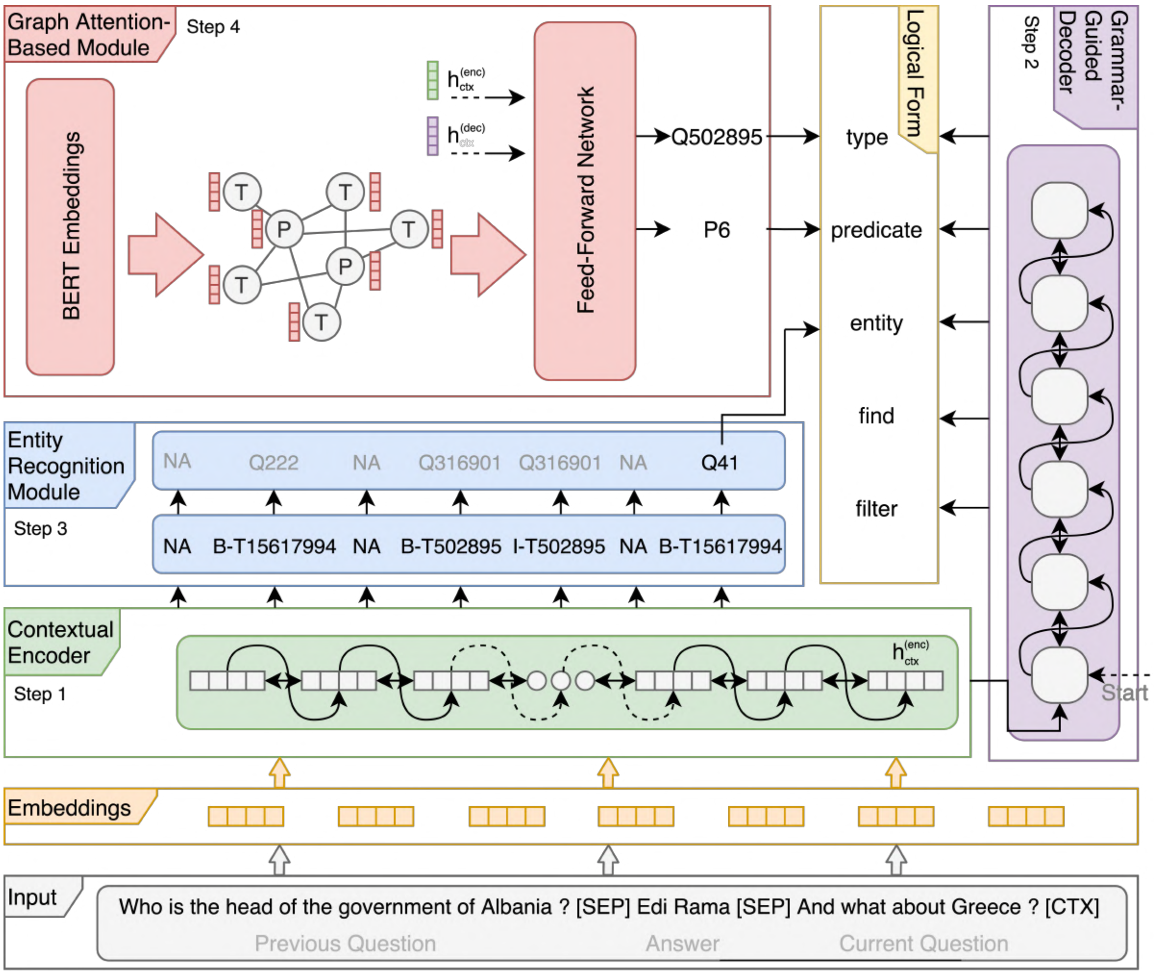

Figure 5.3: LASAGNE (Multi-task Semantic Parsing with Transformer and Graph Attention Networks) architecture. It consists of three modules: 1) A semantic parsing-based Transformer model, containing a contextual encoder and a grammar-guided decoder using the grammar defined in Table 5.6. 2) An entity recognition module identifies all the entities in the context together with their types and links them to the knowledge graph. It filters them based on the context and permutes them, in case of more than one required entity. Finally, 3) a graph attention-based module that uses a GAT network initialised with BERT embeddings to incorporate and exploit correlations between (entity) types and predicates. The resulting node embeddings, together with the context hidden state ($h_{ctx}^{(enc)}$) and decoder hidden state ($h^{(dec)}$), are used to score the nodes and predict the corresponding type and predicate.

Action	Description
set → find(e, p)	set of objects part of the triples with subject e and predicate p
set → find_reverse(e, p)	set of subjects part of the triples with object e and predicate p
set → filter_type(set, tp)	filter the given set of entities based on the given type
set → filter_multi_types(set_1, set_2)	filter the given set of entities based on the given set of types
dict → find_tuple_counts(p, tp_1, tp_2)	extracts a dictionary, where keys are entities of $type_1$ and values are the number of objects of $type_2$ related with p
dict → find_reverse_tuple_counts(p, tp_1, tp_2)	extracts a dictionary, where keys are entities of $type_1$ and values are the number of subjects of $type_2$ related with p
set → greater(dict, num)	set of those entities that have greater count than *num*
set → lesser(dict, num)	set of those entities that have lesser count than *num*
set → equal(dict, num)	set of those entities that have equal count with *num*
set → approx(dict, num)	set of those entities that have approximately same count with *num*
set → atmost(dict, num)	set of those entities that have at most same count with *num*
set → atleast(dict, num)	set of those entities that have at least same count with *num*
set → argmin(dict)	set of those entities that have the most count
set → argmax(dict)	set of those entities that have the least count
boolean → is_in(entity, set)	check if the entity is part of the set
number → count(set)	count the number of elements in the set
set → union(set_1, set_2)	union of set_1 and set_2
set → intersection(set_1, set_2)	intersection of set_1 and set_2
set → difference(set_1, set_2)	difference of set_1 and set_2

Table 5.6: Predefined grammar with respective actions to generate logical forms.

Grammar

For the semantic parsing task, we employ a grammar that can be used to capture the entire context of the input utterance with the minimum number of actions. Table 5.6 illustrates the complete grammar with all the defined actions. We integrated most actions from our previous work (Section 5.3) [14]; however, we have updated particular actions for optimal performance. For instance, we removed the action "*is_subset*" and introduced two new actions such as "*atleast*" and "*atmost*" for handling more complicated scenarios of comparative and quantitative question types.

Transformer

To translate the input conversation into a sequence of actions (logical form), we utilise a Transformer model [82]. Specifically, the Transformer here aims to map a question q, that is a sequence $x = \{x_1, \ldots, x_n\}$, to the label l, that can be also defined as a sequence $y = \{y_1, \ldots, y_m\}$, by modelling the conditional probability $p(y|x)$.

Input and Word Embedding. We have to incorporate the dialog history from previous interactions as an additional input to our model for handling co-reference and ellipsis. To do so, we consider the following utterances for each turn: 1) the previous question, 2) the previous answer, and 3) the current question. Utterances are separated from one another by using a $[SEP]$ token. At the end of the last utterance, we append a context token $[CTX]$, which is used as the semantic representation for the entire input question. In the next step, given an utterance q containing n words $\{w_1, \ldots, w_n\}$ we first tokenise the conversation context using WordPiece tokenization [188], and after that, we use the pre-trained model GloVe [162] to embed the words into a vector representation space of dimension d. Our word embedding model provides us with a sequence $x = \{x_1, \ldots, x_n\}$ where x_i is given by, $x_i = GloVe(w_i)$ and $x_i \in \mathbb{R}^d$.

Contextual Encoder. The word embeddings x, are forwarded as input to the contextual encoder, which uses the multi-head attention mechanism described by [82]. The encoder here outputs the contextual embeddings $h^{(enc)} = \{h_1^{(enc)}, \dots, h_n^{(enc)}\}$, where $h_i^{(enc)} \in \mathbb{R}^d$ and it can be defined as:

$$h^{(enc)} = encoder(x; \theta^{(enc)}), \tag{5.9}$$

where $\theta^{(enc)}$ are the encoder's trainable parameters.

Grammar-Guided Decoder. We use a grammar-guided decoder for generating the logical forms. The decoder also employs the multi-head attention mechanism. The decoder output is dependent on the encoder contextual embeddings $h^{(enc)}$. The main task of the decoder is to generate each corresponding action, based on Table 5.6, alongside with the general semantic category from the knowledge graph (entity, type, predicate). In other words, the decoder will predict the main logical form without using or initialising any specific information from the knowledge graph. Here we define the decoder vocabulary as $V^{(dec)} = \{find, find\_reverse, \dots, entity, type, predicate, value\}$, where all the actions from Table 5.6 are included. On top of the decoder stack, we employ a linear layer alongside a softmax to calculate each token's probability scores in the vocabulary. We define the decoder stack output as follows:

$$\begin{aligned} h^{(dec)} &= decoder(h^{(enc)}; \theta^{(dec)}), \\ p_t^{(dec)} &= softmax(\mathbf{W}^{(dec)} h_t^{(dec)}), \end{aligned} \tag{5.10}$$

where $h_t^{(dec)}$ is the hidden state in time step t, $\theta^{(dec)}$ are the decoder trainable parameters, $\mathbf{W}^{(dec)} \in \mathbb{R}^{|V^{(dec)}| \times d}$ are the linear layer weights, and $p_t^{(dec)} \in \mathbb{R}^{|V^{(dec)}|}$ is the probability distribution over the decoder vocabulary in time step t. $|V^{(dec)}|$ denotes the decoder's vocabulary size.

Entity Recognition Module

The entity recognition module is composed of two sub-modules, where each module is trained using a different objective.

Entity Detection and Linking

Entity Detection. It aims to detect and link the entities to the KG. The module is inspired by [13] and performs type-aware entity detection by using BIO sequence tagging jointly with entity type tagging. Specifically, the entity detection vocabulary is defined as $V^{(ed)} = \{O, \{B, I\} \times \{TP_i\}_{i=1}^{N^{(tp)}}\}$, where TP_i denotes the *i-th* entity type label, $N^{(tp)}$ stands for the number of the distinct entity types in the knowledge graph and $|V^{(ed)}| = 2 \times N^{(tp)} + 1$. For performing the sequence tagging task we use an LSTM [79] and the module is defined as:

$$\begin{aligned} h^{(l)} &= LeakyReLU(LSTM(h^{(enc)}; \theta^{(l)})), \\ p_t^{(ed)} &= softmax(\mathbf{W}^{(l)} h_t^{(l)}), \end{aligned} \tag{5.11}$$

where $h^{(enc)}$ is the encoder hidden state, $\theta^{(l)}$ are the LSTM layer trainable parameters, $h_t^{(l)}$ is the LSTM hidden state for time step t, $\boldsymbol{W}^{(l)} \in \mathbb{R}^{|V^{(ed)}| \times d}$ are the linear layer weights and $p_t^{(ed)}$ are the entity detection module prediction for time step t. $|V^{(ed)}|$ denotes the entity detection vocabulary size.

Entity Linking. Once the entity BIO labels and their types are recognised, the next steps for the entity linking are: 1) the BIO labels are used to locate the entity spans from the input utterances. 2) An inverted index built for the knowledge graph entities is used to retrieve candidates for each predicted entity span. Finally, 3) the candidate lists are filtered using the predicted (entity) types. From the filtered candidates, the first entity is considered as correct.

Filtering and Permutation

After finding all the input utterances' entities, we perform two additional tasks in order to use entities in the generated logical form. First, we filter the relevant entities, and then we need to permute the entities in the order required for the logical form. The module receives as an input the concatenation of the hidden states of the encoder $h^{(enc)}$ and the hidden states of the LSTM $h^{(l)}$ from the entity detection model. The module here learns to assign index tags to each input token. We define the module vocabulary as $V^{(ef)} = \{0, 1, \ldots, m\}$ where 0 is the index assigned to the context entities that are not considered. The remaining values are indices that permute our entities based on the logical form. Here, m is the total number of indices based on the maximum number of entities from all logical forms. Overall, our filtering and permutation module is modelled using a feed-forward network with two linear layers separated with a Leaky ReLU activation function and appended with a softmax. Formally we define the module as:

$$\begin{aligned} h^{(ef)} &= LeakyReLU(\boldsymbol{W}^{(ef_1)}[h^{(enc)}; h^{(l)}]), \\ p_t^{(ef)} &= softmax(\boldsymbol{W}^{(ef_2)} h_t^{(ef)}), \end{aligned} \tag{5.12}$$

where $\boldsymbol{W}^{(ef_1)} \in \mathbb{R}^{d \times 2d}$ are the weights of the first linear layer and $h_t^{(ef)}$ is the hidden state of the module in time step t. $\boldsymbol{W}^{(ef_2)} \in \mathbb{R}^{|V^{(ef)}| \times d}$ are the weights of the second linear layer, $|V^{(ef)}|$ is the size of the vocabulary and $p_t^{(ef)}$ denotes the probability distribution over the tag indices for the time step t.

Graph Attention-Based Module

A knowledge graph (KG) can be denoted as a set of triples $\mathcal{K} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ where $\mathcal{E}$ and $\mathcal{R}$ are the set of entities and relations respectively. To build the (local) graph, we consider the relations and the types of entities that are linked with these relations in the knowledge graph $\mathcal{K}$. We define a graph $\mathcal{G} = \{\mathcal{T} \cup \mathcal{R}, \mathcal{L}\}$ where $\mathcal{T}$ is the set of types, $\mathcal{R}$ is the set of relations and $\mathcal{L}$ is a set of links (tp_1, r) and (r, tp_2), such that $\exists (e_1, r, e_2) \in \mathcal{K}$ where e_1 is of type tp_1 and e_2 is of type tp_2.

To propagate information in the graph and to project prior KG information into the embedding space, we use the Graph Attention Networks (GATs) [107].

Figure 5.4 shows the aggregation process of graph attention layer between the (entity) types and predicates from Wikidata. The KG types and predicates are the nodes of the graph, and there exist an

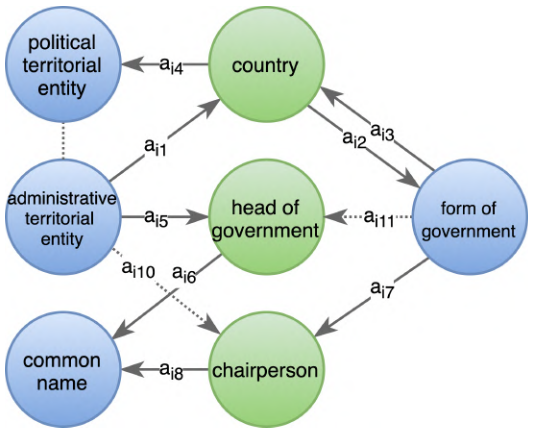

Figure 5.4: The aggregation process of graph attention layer between the (entity) types and predicates from Wikidata knowledge graph. The dashed lines represent an auxiliary edge, while a_{ij} represents relative attention values of the edge. We also incorporate the predicates (relations) as nodes of the graph instead of edges.

edge only between types and predicates with the condition that there exist a triple which involved the predicate and an entity of that type. We use GATs [107] to capture different level of information for a node, based on the neighborhood in the graph. We denote with $h^{(g)} = \{h_1^{(g)}, \ldots, h_n^{(g)}\}$ the initial representations of the nodes, which will also be the input features for the GAT layer. To denote the influence of node j to the node i, an attention score e_{ij} is computed as $e_{ij} = a(\mathbf{W}h_i^{(g)}, \mathbf{W}h_j^{(g)})$, where $\mathbf{W}$ is a parameterized linear transformation, and a is an attention function. In our case, we follow the GAT work [107], and compute e_{ij} score as follows,

$$e_{ij} = LeakyReLU(\mathbf{a}^T[\mathbf{W}h_i^{(g)} || \mathbf{W}h_j^{(g)}]), \tag{5.13}$$

where $\mathbf{a} \in \mathbb{R}^{2d}$ is a single-layer feedforward network, and $||$ denotes concatenation. This attention scores are normalized using a softmax function and producing the α_{ij} scores for all the edges in a neighborhood. These normalized attention scores are used to compute the output features $\overline{h}_i^{(g)}$ of a node in a graph, by applying a linear combination of all the nodes in the neighborhood as below,

$$\overline{h}_i^{(g)} = \sigma(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W} h_j^{(g)}), \tag{5.14}$$

where σ is a non-linear function. Following [107] and [82] we also apply a multi-head attention mechanism and compute the final output features as,

$$\overline{h}_i^{(g)} = \sigma(\frac{1}{K} \sum_{k=1}^{K} \sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k h_j^{(g)}), \tag{5.15}$$

where K is equal to the number of heads, and α_{ij}^k, $\mathbf{W}^k$ are the corresponding attention scores and linear transformation by the k-th attention mechanism. During our experiments, we found out that $K = 2$ was sufficient for our model.

For our task, we initialise each node embedding $h^{(g)} = \{h_1^{(g)}, \ldots, h_n^{(g)}\}$ using pretrained BERT embeddings, and $n = |\mathcal{T} \cup \mathcal{R}|$. A GAT layer uses a parameter weight matrix, and self-attention, to produce a transformation of input representations $\overline{h}^{(g)} = \{\overline{h}_1^{(g)}, \ldots, \overline{h}_n^{(g)}\}$, where $\overline{h}_i^{(g)} \in \mathbb{R}^d$ as shown below:

$$\overline{h}^{(g)} = g(h^{(g)}; \theta^{(g)}),^{6} \tag{5.16}$$

where $g(\cdot)$ performs the GAT process described above and $\theta^{(g)}$ are the trainable parameters. We model the task of predicting the correct type or predicate in the logical form as a classification task over the nodes in graph $\mathcal{G}$, given the current conversational context and decoder hidden state. For each time step t in the decoder, we calculate the probability distribution $p_t^{(g)}$ over the graph nodes as:

$$p_t^{(g)} = softmax(\overline{h}^{(g)T} h_t^{(c)}), \tag{5.17}$$

where $\overline{h}^{(g)} \in \mathbb{R}^{d \times n}$ and $h_t^{(c)}$ is a linear projection of the concatenation of the context representation and the decoder hidden state, given as follows,

$$h_t^{(c)} = LeakyReLU(\boldsymbol{W}^{(g)}[h_{ctx}^{(enc)}; h_t^{(dec)}]), \tag{5.18}$$

and $\boldsymbol{W}^{(g)} \in \mathbb{R}^{d \times 2d}$.

5.4.2 Multi-Task Learning

The framework consists of four trainable modules, the grammar-guided decoder, entity detection, filtering and permutation, and GAT-based module for types and predicates. Every module consists of a loss function that contributes to the overall performance of the framework, as shown in Section 5.4.5. To account for multi-tasking, we perform a weighted average of all the single losses:

$$L = \lambda_1 L^{dec} + \lambda_2 L^{ed} + \lambda_3 L^{ef} + \lambda_4 L^{g}, \tag{5.19}$$

where $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ are the relative weights, which are learned during training by taking into account the difference in magnitude between losses by incorporating the log standard deviation [147, 165]. L^{dec}, L^{ed}, L^{ef}, and L^{g} are the respective negative log-likelihood losses of the grammar-guided decoder, entity detection, filtering and permutation, and GAT-based modules. These losses are defined as follows:

$$\begin{aligned}
L^{dec} &= -\sum_{k=1}^{m} logp(y_k^{(dec)}|x), \\
L^{ed} &= -\sum_{j=1}^{n} logp(y_j^{(ed)}|x), \\
L^{ef} &= -\sum_{i=1}^{n} logp(y_i^{(ef)}|x), \\
L^{g} &= -\sum_{k=1}^{m} I_{(y_k^{(dec)} \in \{type, pred\})} logp(y_k^{(g)}|x),
\end{aligned} \tag{5.20}$$

[6] For more details about GAT please refer to the appendix.

where n and m are the length of the input utterance x and the gold logical form, respectively. $y_k^{(dec)} \in V^{(dec)}$ are the gold labels for the decoder, $y_j^{(ed)} \in V^{(ed)}$ are the gold labels for entity detection, $y_j^{(ef)} \in V^{(ef)}$ are the gold labels for filtering and permutation, and $y_k^{(g)} \in \{\mathcal{T} \cup \mathcal{R}\}$ are the gold labels for the GAT-based module. The model benefits from multiple supervision signals from each module, and this improves the performance in the given task.

5.4.3 Experimental Setup

In this subsection, we provide the setup for the experiments. We further describe the resources and metrics we employ.

Datasets

We use the Complex Sequential Question Answering (CSQA) dataset[7] [117]. CSQA was built on the large-scale knowledge graph Wikidata. Wikidata consists of 21.2M triples with over 12.8M entities, 3,054 entity types, and 567 predicates. The CSQA dataset consists of around 200K dialogues where each partition – train, valid, test contains 153K, 16K, 28K dialogues, respectively. The questions involve complex reasoning to determine the correct answers.

Model Configurations

We follow a similar approach to our previous work (Section 5.3) to annotate the CSQA dataset with gold logical forms. For the Transformer module, we use the configurations from [82]. Our model dimension is $d = 300$, with a total number of $H = 6$ heads and $L = 2$ layers. The inner feed-forward linear layers have dimension $d_{ff} = 600$. Following the base Transformer parameters, we apply residual dropout [167] to the summation of the embeddings and the positional encodings in both encoder and decoder stacks with a rate of 0.1. The entity detection module has a dimension of 300. The filtering and permutation module receives an input of dimension 600 where here a linear layer is responsible to reduce it to 300 which is the framework dimension. For the GAT-based module, as mentioned above, we use pre-trained BERT embeddings for type and predicate labels. Hence the input dimension on this module is 3072[8]. The GAT layer will produce representations with an embedding size of 300. For the optimisation, we use the Noam optimiser proposed by [82], where authors use an Adam optimiser [166] with several warmup steps for the learning rate.

Models for Comparison

We compare the LASAGNE framework with the last three baselines that have been evaluated on the employed dataset. The first baseline is a HRED+KVmem model [117]. The second baseline is D2A [12], which uses a semantic parsing approach based on a seq2seq model. Finally, the previous state-of-the-art is MaSP [13], which is also a semantic parsing approach. Please note, the number of parameters for LASAGNE were 14.7M compared to MaSP with 15M. Our base Transformer model can be replaced with larger models like BERT with extremely large number of parameters for performance gain, however, that was not the focus of this work.

[7] https://amritasaha1812.github.io/CSQA

[8] We concatenate the last four layers of BERT where each layer has a dimension of 768.

Evaluation Metrics

We use the same metrics as employed by the authors of the CSQA dataset [117] as well as the previous baselines. The F1-score is used for questions that have an answer composed of a set of entities. The Accuracy metric is used for the question types whose answer is a number or a boolean value (YES/NO). We also provide an overall score for each evaluation metric and their corresponding question categories.

5.4.4 Results

Table 5.7 summarises the results comparing the LASAGNE framework against the previous baselines. LASAGNE outperforms the previous baselines weighted average on all question types (The row "overall" in the Table 5.7). Furthermore, LASAGNE is a new SotA in 8 out of 10 question types, and in some cases, the improvement is up to 31 percent.

Methods		HRED-KVM	D2A	MaSP	LASAGNE (ours)	Δ
# Train Param		-	-	15M	14.7M	
Question Type	**#Examples**	F1 Score				
Overall	206k	9.39%	66.70%	79.26%	**82.91%**	+3.65%
Clarification	12k	16.35%	35.53%	**80.79%**	69.46%	-11.33%
Comparative Reasoning (All)	15k	2.96%	48.85%	68.90%	**69.77%**	+0.87%
Logical Reasoning (All)	22k	8.33%	67.31%	69.04%	**89.83%**	+20.79%
Quantitative Reasoning (All)	9k	0.96%	56.41%	73.75%	**86.67%**	+12.92%
Simple Question (Coreferenced)	55k	7.26%	57.69%	76.47%	**79.06%**	+2.59%
Simple Question (Direct)	82k	13.64%	78.42%	85.18%	**87.95%**	+2.77%
Simple Question (Ellipsis)	10k	9.95%	81.14%	**83.73%**	80.09%	-3.64%
Question Type	**#Examples**	Accuracy				
Overall	66k	14.95%	37.33%	45.56%	**64.34%**	+18.78%
Verification (Boolean)	27k	21.04%	45.05%	60.63%	**78.86%**	+18.23%
Quantitative Reasoning (Count)	24k	12.13%	40.94%	43.39%	**55.18%**	+11.79%
Comparative Reasoning (Count)	15k	8.67%	17.78%	22.26%	**53.34%**	+31.08%

Table 5.7: LASAGNE's performance comparison on the CSQA dataset having 200K dialogues with 1.6M turns and over 12.8M entities. LASAGNE achieves "overall" (weighted average on all question types) new state-of-the-art for both the F1 score and the question type results' accuracy metric.

What worked in our case? For question types that require more than two entities for reasoning, such as *Logical Reasoning (All)* and *Verification (Boolean)*, LASAGNE performs considerably better (+20.79% and +18.23% respectively). This is mainly due to the proposed entity recognition module. Furthermore, for question types that require two or more (entity) types and predicates, such as *Quantitative Reasoning (All)*, *Quantitative Reasoning (Count)* and *Comparative Reasoning (Count)* LASAGNE also outperforms MaSP (+12.92%, +11.79% and +31.08% respectively). Here, the improvement is due to the graph attention-based module, which is responsible for predicting the relevant (entity) types and predicates. Another interesting result is that LASAGNE also performs better in two out of three *Simple Question involving one entity and one predicate* categories. The performance shows the robustness of LASAGNE.

What did not work in our case? LASAGNE noticeably under-performs on the *Clarification* question type, where MaSP retains the state-of-the-art. The main reason is the spurious logical forms during the annotation process which has further impacted the *Simple Questions (Ellipsis)* performance.

5.4.5 Ablation Study

Effect of GAT and Multi-task Learning. Table 5.8 summarises the effectiveness of the GAT-based module and the multi-task learning. We can observe the advantage of using them together in LASAGNE. To show the effectiveness of GAT-based module, we replace it with two simple classifiers, one for each predicate and type categories. We can observe that the performance drops significantly for the question types that require multiple entity types and predicates (e.g. *Quantitative Reasoning (All)*, *Quantitative Reasoning (Count)* and *Comparative Reasoning (Count)*). When we exclude the multi-task learning and train all the modules independently, there is a negative impact on all question types. In LASAGNE, the filtering and permutation module, along with the GAT-based module, is heavily dependent on the supervision signals received from the previous modules. Therefore it is expected that without the multi-task learning, LASAGNE will underperform on all question types, since each module has to re-learn inherited information.

Methods	Ours	w/o GATs	w/o MTL
Question Type		F1 Score	
Clarification	66.94%	57.33%	59.43%
Comparative	69.77%	57.72%	66.41%
Logical	89.83%	78.52%	86.75%
Quantitative	86.67%	75.26%	82.18%
Simple (Coref)	79.06%	76.46%	77.23%
Simple (Direct)	87.95%	83.59%	85.39%
Simple (Ellipsis)	80.09%	77.19%	78.47%
Question Type		Accuracy	
Verification	78.86%	63.38%	75.24%
Quantitative	55.18%	40.87%	46.27%
Comparative	53.34%	41.73%	45.90%

Table 5.8: The effectiveness of the GAT and the multi-task learning. The first column contains the results of the LASAGNE framework, where all the modules are trained simultaneously. The second and third columns selectively remove the GAT and the multi-task learning from LASAGNE.

5.4.6 Task Analysis

Table 5.9 illustrates the task accuracy of LASAGNE. The Entity Detection task has the lowest accuracy (86.75%). The main reason here is the errors in the entity type prediction. On the other hand, for all other tasks, we have accuracy above 90%.

Effect of Filtering and Permutation. For justifying the effectiveness and superior performance of LASAGNE's filtering and permutation module, we compare the overall performance of the entity

Tasks	Accuracy
Entity Detection	86.75%
Filtering & Permutation	97.49%
Grammar-Guided Decoder for Logical Forms	98.61%
GAT-Based Module for Type/Predicate	92.28%

Table 5.9: Tasks accuracy of the LASAGNE framework.

Model	Entity Recognition Accuracy
MaSP	79.8%
LASAGNE	92.1%

Table 5.10: Comparing MaSP [13] and LASAGNE for entity recognition performance.

recognition module to the corresponding module from MaSP. Please note, entity detection modules in both frameworks adopt a similar approach as defined in subsection 5.4.1. In Table 5.10 we can see that the MaSP entity recognition module provides an overall accuracy of 79.8% on test data, while our module outperforms it with an accuracy of 92.1%. The main reason for the under-performance of MaSP is that it uses only token embeddings without any entity information. In contrast, our approach avoids re-learning entity information in the question context and relies on the entity detection module's information.

5.4.7 Error Analysis

For the error analysis, we randomly sampled 100 incorrect predictions. We detail the reasons for two types of errors observed in the analysis:

Entity Ambiguity. Even though our entity detection module assigns (entity) types to each predicted span, entity ambiguity remains the biggest challenge for our framework. For instance, for the question, "Who is associated with Jeff Smith ?" LASAGNE entity detection module correctly identifies "Jeff Smith" as an entity surface form and correctly assigns the (entity) type "common name". However, the Wikidata knowledge graph contains more than ten entities with exactly the same label and type. Our entity linking module has difficulties in such cases. Wikidata entity linking is a newly emerging research domain that has its specific challenges such as entities sharing the same labels, user-created non-standard entity labels and multi-word entity labels (up to 62 words) [189]. Additional entity contexts, such as entity descriptions and other KG contexts, could help resolve the Wikidata entity ambiguity [40].

Spurious Logical Form. For specific question categories, we could not identify gold actions for all utterances. Therefore spurious logical form is a standard error that affects LASAGNE. Specifically, we have spurious logical forms for categories such as "Comparative, Quantitative, and Clarification" but still can achieve SotA in the comparative and quantitative categories.

5.4.8 Synopsis

In this work, we provide a Transformer-based framework to handle the conversational question answering task in a multi-task semantic parsing manner. At the same time, we propose a named entity recognition module for entity detection, filtering, and permutation. Furthermore, we also introduce a graph attention-based module, which exploits correlations between (entity) types and predicates for identifying the gold ones for each particular context. We empirically show that our model achieves the best results for numerous question types and overall. Our ablation study demonstrates the effectiveness of multi-task learning and our graph-based module. We also present an error analysis on a random sample of "*wrong examples*" to discuss our model's weaknesses.

5.5 Summary

In this chapter, we proposed two approaches, CARTON and LASAGNE, for conversational question answering over knowledge graphs. They both handle the task vie multi-task neural semantic parsing, where they aim to generate a logical form and execute it over a KG to retrieve answers. They employ a Transformer model for generating the ground logical form and operate with similar grammars.

The first one (CARTON) uses stacked pointer networks to identify relevant KG information and fill the logical form. The stacked pointer networks consist of three-pointer networks where each one is responsible for each semantic category in the KG (i.e., entities, relations, and types). However, CARTON requires the list of mentioned KG entities in the conversation to operate since the model will only handle identifying which of those entities are relevant and where should be placed in the logical form. Therefore, CARTON does not perform entity detection and entity linking tasks. The model's novelty is found in the semantic grammar, the stacked pointer networks, and the sequential order of the modules in the overall architecture.

The second approach (LASAGNE) is architecturally more complex than CARTON (also a higher number of training parameters). However, it is complete since it performs all the required tasks to answer conversational questions without relying on external information (e.g., the list of KG entities). This model fills the logical forms using two novel sub-modules: i) an entity recognition module that, given the input conversation, detects all the entities and links them with the underlying KG. Next, it filters the entities to identify only the relevant ones required for the logical form. In the end, it permutes the filtered list to match the order of the logical form positions (avoiding misplacing entities). ii) A graph attention-based module that exploits the correlations between (entity) types and predicates due to its message-passing ability between the nodes. Motivated from the idea that all selected (entity) types and relations should be connected in the KG.

To conclude, both approaches are novel in their way and achieve state-of-the-art results in one of the most challenging and competitive conversational QA resources.

CHAPTER 6

Conversational Question Answering with Answer Verbalization

In the previous chapter (Chapter 5), we developed multi-task neural semantic parsing systems for conversational question answering. We introduced a new semantic grammar and leveraged state-of-the-art deep neural architectures to incorporate relevant KG information for addressing and overcoming various weaknesses identified in baseline approaches. For the approaches in the previous chapter, we employed conversational context only from the last interaction with the user. However, the entire dialog history may be required to answer conversational questions since the user might refer to information from any previous interaction. Therefore, conversational context plays a vital role in enhancing ConvQA performance. Here, we aim to identify the available context and effectively incorporate it to improve the performance. On the grounds of this, verbalized answers can provide context that might be helpful for particular conversational scenarios. However, it is challenging to incorporate the answer verbalization task into the ConvQA task since we have to determine how to establish a connection between the sub-tasks for sharing training signals and controlling the learning process for optimal performance.

In this chapter, we aim to go a step further and incorporate answer verbalization in the ConvQA task. More precisely, we address two relatively unexplored tasks—first, the ranking-based ConvQA task for answering conversational questions posed against a knowledge graph. Second, the answer verbalization task for generating fluent answer responses while maintaining grammatical correctness. We believe that answer verbalization provides fluent and unambiguous responses to users and supplies additional textual context for improving the ConvQA task performance. For jointly learning the tasks, we utilize multi-task learning.

We address the following research question in this chapter:

RQ4: How can answer verbalization be leveraged to improve the performance of conversational question answering?

Contributions of this chapter are summarized as follows:

- We propose a ConvQA over KGs approach that jointly models the available conversational context (full dialog history with verbalized answers) and KG paths in a common space for learning joint embedding representations to improve KG path ranking.

- We extend the existing ConvQA datasets [16, 17] with verbalized answers.
- We systematically study the impact of incorporating additional context on the ConvQA performance. Results on standard datasets show a considerable improvement over previous baselines. Our evaluation results establish a new baseline, which we believe will drive future research in a new way of developing such frameworks.

This chapter is based on the following publication:

- **Endri Kacupaj**, Kuldeep Singh, Maria Maleshkova, and Jens Lehmann. "Contrastive Representation Learning for Conversational Question Answering over Knowledge Graphs." In Proceedings of the 31st ACM International Conference on Information & Knowledge Management, pp. 925-934. 2022. DOI: 10.1145/3511808.3557267

The structure of the chapter is as follows: Section 6.1 introduces the work. Section 6.2 presents the related work. Section 6.3 provides the concepts, notations and formulates the task. Section 6.4 presents the proposed approach. Section 6.5 describes the multi-task learning process. Section 6.6 shows how we extended existing ConvQA datasets with answer verbalization. Section 6.7 illustrates the experimental setup, while Section 6.8 the experiments results. Section 6.9 provides a detailed ablation study. A task analysis is on Section 6.10. An error analysis for ConvQA and answer verbalization tasks is on Section 6.11. We summarize in Section 6.12.

6.1 Introduction

For the KGQA setup, the existing scientific-literature can be broadly classified into two categories [145, 190, 191]: i) semantic parsing approaches, where the goal is to map questions into a logical form, which is then executed over the knowledge graph to extract the correct answers. ii) Information retrieval approaches aim to retrieve a question-specific graph and apply ranking algorithms to select entities for top positions. The two approaches follow either a parse-then execute paradigm or a retrieval-and-rank paradigm. For ConvQA over KGs, there has been significant progress on semantic parsing-based approaches [13–15]. However, collecting training data for semantic parsing approaches is challenging, and time-consuming [5] since each question must be associated with a gold logical form. While for the information-retrieval/ranking-based approaches, only the correct answers (e.g., entities) are required for each question.

Existing ranking-based ConvQA techniques are restricted to only generating or producing answers without verbalizing them in natural language [190]. The lack of verbalization makes the interaction with the user unnatural and often leaves the users with ambiguity [192]. For example, for the first question in Figure 6.1, "What countries did the main character travel in the book Eat, Pray, Love?" existing ConvQA systems [13–17] will respond with the countries as an answer with no further explanation. In such cases, the user might need to refer to external data sources to verify the answer. A verbalized answer, as common in task-oriented dialogues [193], would allow the user to confirm that the answer is related to the context since it also includes additional characteristics that indicate how it was determined. Furthermore, in contrast to task-oriented dialog systems [193, 194], existing ranking-based ConvQA approaches [16, 17] do not consider the entire dialog history while seeking an answer of a given question in the specific turn of the dialog for a particular domain (cf. Figure 6.1).

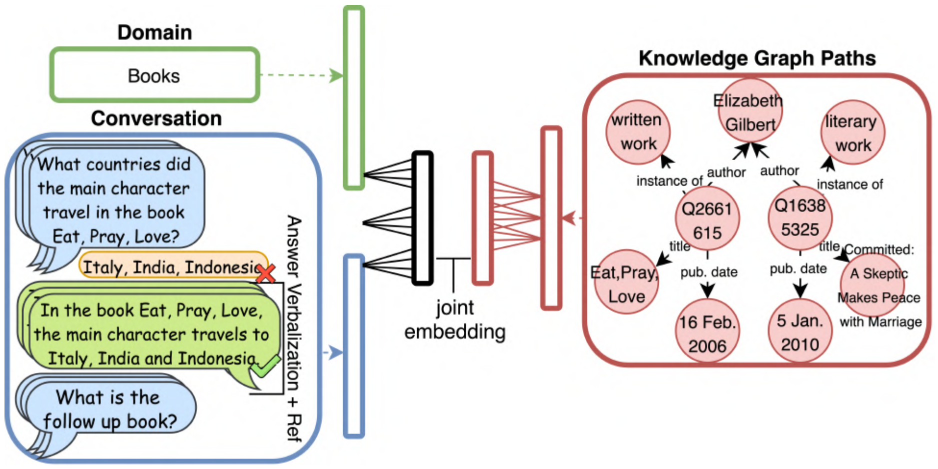
Domain
Books
Conversation
What countries did the main character travel in the book Eat, Pray, Love?
Italy, India, Indonesia
In the book Eat, Pray, Love, the main character travels to Italy, India and Indonesia
What is the follow up book?
Answer Verbalization + Ref
joint embedding
Knowledge Graph Paths
written work
Elizabeth Gilbert
literary work
instance of
author
author
instance of
Q2661615
Q16385325
title
title
Eat,Pray, Love
pub. date
pub. date
Committed: A Skeptic Makes Peace with Marriage
16 Feb. 2006
5 Jan. 2010

Figure 6.1: Motivating example illustrating a sample conversation [16]. For conversational question answering over KGs, availability of entire dialog history, domain information, and verbalized answers may act as sources of context in determining the ranking of KG paths while retrieving correct answers. Our proposed approach models conversational context and KG paths in a common space by jointly learning the embeddings for homogeneous representation.

Hence, we hypothesize that incorporating the available conversational context, i.e. (1) the entire dialog history with (2) verbalized answers and (3) domain information, may positively impact the overall performance of a ConvQA approach while retrieving answers from the underlying KG. Combination of these three sources of context has not been exploited in the literature to improve ConvQA. Our rationale for this is: 1) Current ConvQA approaches only incorporate context from the previous turn without verbalized answers [15–17]. 2) Considering conversational context plays a vital role in human understanding [195] and question answering [196], verbalized answers in the conversational history provide more textual context than the simple retrieved answers. This supplementary knowledge may play a crucial role in learning more efficient joint embeddings of conversation context and KG-paths to impact the final ranking results. 3) Domain information will filter less relevant KG paths for a given conversation. It is due to the fact that some KG relations are more likely to occur in particular domains.

In this context, we propose PRALINE (**P**ath **R**anking for convers**A**tiona**L** quest**I**on a**N**sw**E**ring), the first ConvQA over KGs approach that tackles the QA task through a path ranking approach and at the same time fluently verbalizes answer responses. PRALINE consists of four modules trained simultaneously. The first module encodes the input conversation, while the next two modules jointly learn to embed conversations and KG paths in a common space, which is semantically regularized by adding the domain identification task. The last module is responsible for verbalizing the retrieved answers to provide additional textual context for the KG-path ranking. To facilitate reproducibility and reuse, our framework implementation, and the extended dataset with verbalized answers are publicly available[1].

[1] https://github.com/endrikacupaj/PRALINE

6.2 Related Work

Considering KGQA is a widely studied research topic, we stick to the work closely related to our proposed approach (detailed surveys are in [4, 190]).

Single-shot KGQA

Several KGQA works handle the task as a semantic graph generation and re-ranking. Bast et al. [197] compare a set of manually defined query templates against the natural language question and generate a set of query graph candidates by enriching the templates with potential relations. Yih et al. [198] creates grounded query graph candidates using a staged heuristic search algorithm and employs a neural ranking model to score and find the optimal semantic graph. Yu et al. [199] use a hierarchical representation of KG relations in a neural query graph ranking model. Authors compare the results against a local sub-sequence alignment model with cross attention [200]. Maheshwari et al. [201] conduct an empirical investigation of neural query graph ranking approaches by experimenting with six different ranking models. The proposed approach is a self-attention-based slot matching model that exploits the inherent structure of query graphs.

ConvQA over KGs

Most recent works on ConvQA [13–15] employ the semantic parsing approach to answer conversational questions. The first work in this area [117] propose a hybrid model of the HRED model [179] and the key-value memory network model [180]. The model consists of three components; where the first one is the hierarchical encoder, which computes the utterance representation. The next module is a higher-level encoder that computes the context representation. The second component is the Key-Value Memory Network that stores each candidate tuples as a key-value pair. The key contains the concatenated embedding of the relation and the subject. The last component is the decoder used to create an end-to-end solution and produce multiple types of answers. Christmann et al. [16] proposes an approach that answers conversational questions over a knowledge graph (KG) by maintaining conversation context using entities and predicates seen so far and automatically inferring missing or ambiguous pieces for follow-up questions. The core of this method is a graph exploration algorithm that judiciously expands a frontier to find and rank candidate answers for the given question. Kaiser et al. [17] present a reinforcement learning model that can learn from a conversational stream of questions and reformulations. Authors model the answering process as multiple agents walking in parallel on the KG, where the walks are determined by actions sampled using a policy network. The policy network takes the question and the conversational context as inputs and is trained via noisy rewards obtained from the reformulation likelihood. Our work lies closely with [16, 17]. However, these approaches ignore the full dialog history and do not consider verbalized answers as the contextual sources. Our focus is to explore the impact of full conversational context on KG path ranking to retrieve final answers.

6.3 Concepts, Notation and Problem Formulation

We define a KG as a tuple $\mathcal{K} = (\mathcal{E}, \mathcal{R}, \mathcal{T}^+)$ where $\mathcal{E}$ denotes the set of entities (vertices), $\mathcal{R}$ is the set of relations (edges), and $\mathcal{T}^+ \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is a set of all triples. A triple $\tau = (e_h, r_{h,t}, e_t) \in \mathcal{T}^+$ indicates that, for the relation $r_{h,t} \in \mathcal{R}$, e_h is the head entity (origin of the relation) while e_t is the tail entity. For a KG, a conversation C with T turns is composed from a set of a sequence of questions $Q = \{q^t\}$ and corresponding answers $\mathcal{A} = \{a^t\}$, where $t = 0, 1, ..., \mathrm{T}$, such that $C = \langle(q^0, a^0), (q^1, a^1), ..., (q^\mathrm{T}, a^\mathrm{T})\rangle$. Furthermore, each question q^t is a sequence of tokens q_i^t, such that $\langle q_1^t, ..., q_{|q^t|}^t\rangle$, where $|q^t|$ is the number of tokens in q^t. For each question q^t we have a conversation history C^t, where for question q^0 the conversation history is $\emptyset$. We define as a^t the answer for each question q^t which is a set of entities or literals in $\mathcal{K}$. We define verbalized answer as v^t, which is a sequence of tokens v_i^t, where $a^t \in v^t$. Similar to question q^t we have $\langle v_1^t, ..., v_{|v^t|}^t\rangle$, where $|v^t|$ is the number of tokens in v^t. With verbalized answers the conversation C can be illustrated as $C = \langle(q^0, v^0), (q^1, v^1), ..., (q^\mathrm{T}, v^\mathrm{T})\rangle$. A notation overview is in Table 6.1.

Notation	Concept
$\mathcal{K}, \mathcal{E}, \mathcal{R}, \mathcal{T}^+$	Knowledge Graph, entities, relations, triples
C, t	Conversation, turn
q^t, a^t	Question and answer at turn t
v^t	Verbalized answer at turn t
τ^t	Domain information at turn t
C^t	Conversation history at turn t
$\mathcal{E}_c, \mathcal{P}_c$	Context entities, context KG paths
$\mathcal{D}^{t+}, \mathcal{D}^{t-}$	Set of positive and negative context paths for q^t
s^t	Input sequence (contains C^t and q^t)
d	Space dimension
$h^{(\cdot)}$	Contextual embeddings
$\theta^{(\cdot)}$	Trainable parameters
$\boldsymbol{W}^{(\cdot)}$	Weight matrix for linear layer
$\omega^{(\cdot)}$	Probability distribution over vocabulary
ϕ^c, ϕ^p	Joint embeddings for conversation and path

Table 6.1: Notation for concepts in PRALINE.

Context Entities

The set of context entities $\mathcal{E}_c \subseteq \mathcal{E}$ contains entities mentioned in question q^t, answer a^t (or verbalized answer v^t) and conversation C^t.

Context Paths

Context KG paths $\mathcal{P}_c$ are extracted given the context entities $\mathcal{E}_c$, where $\mathcal{P}_c \subseteq \{\mathcal{E}_c \times \mathcal{R} \times \mathcal{E}\} + \{\mathcal{E}_c \times \mathcal{R} \times \mathcal{E} \times \mathcal{R} \times \mathcal{E}\} + \{\mathcal{E}_c \times \mathcal{R} \times \mathcal{E} \times \mathcal{R} \times \mathcal{E} \times \mathcal{R} \times \mathcal{E}\}$. This means that extracted KG paths $\mathcal{P}_c$ will be either 1-hop, 2-hop or 3-hop paths where all of them start with context entities $\mathcal{E}_c$. For a question q^t, we define $\mathcal{D}^{t+} \subseteq \mathcal{P}_c^t$ and $\mathcal{D}^{t-} \subseteq \mathcal{P}_c^t$ the set of positive/correct and negative/incorrect context paths, respectively. Where, $\mathcal{D}^{t+} \cup \mathcal{D}^{t-} = \mathcal{P}_c^t$.

Answer Verbalization

For the answer verbalization task, given the question q^t, the conversation history C^t, and the answer a^t, we aim to generate a natural language sentence v^t, with the requirements that it is grammatically sound and correctly represents all the information in the question q^t, conversation C^t, and answer a^t. Formally, let X, Y denote the source-target pair. X contains the set of questions Q, conversations C, answers $\mathcal{A}$, and Y corresponds to set of verbalized answers V. The goal of the answer verbalization is to learn a distribution $p(Y|X)$ to generate natural language text describing the answer automatically.

Problem Formulation

For the ConvQA task, given a knowledge graph $\mathcal{K}$, a natural language question q^t, the conversation history C^t, and the set of context entities $\mathcal{E}_c^t$, we first to extract all the potential KG paths $\mathcal{P}_c^t$ correlated to context entities. We formulate the task as a ranking problem where we score and rank the KG paths $\mathcal{P}_c^t$ in order to select the top context paths $p_c^t \in \mathcal{P}_c^t$ that leads us to entities or literals that belong to the gold answer a^t, which is also the answer of the question q^t.

6.4 Path Ranking for Conversational Question Answering

In a conversation, the input data consists of questions q^t and their answers a^t, extracted from the knowledge graph. We propose a path ranking based approach implemented in the PRALINE framework. In particular, it ranks (KG) context paths $\mathcal{P}_c^t$ depending on the conversation context. Figure 6.2 shows the architecture of PRALINE.

6.4.1 Encoder

As the first step of our framework, we utilize a BART [93] Transformer-based encoder [82] in order to encode both the conversation history C^t and current question q^t at turn t. The conversation history also contains verbalized answers from previous turns (e.g. $v^{t-1}, v^{t-2}, ...$). Here we concatenate the conversation history C^t and current question q^t using a helper token $[SEP]$ to create the input sequence $s^t = C^t \oplus [SEP] \oplus q^t$, where $\oplus$ is the operation of sequence concatenation.[2] Next, we tokenize the input sequence s^t into $|s^t|$ tokens $\{s_1^t, ..., s_{|s^t|}^t\}$, where $|s^t| = |C^t| + |q^t| + 1$, using a byte-level Byte-Pair-Encoding tokenizer [202]. Then, we forward the tokenized sequence into the encoder and it outputs the contextual embeddings $h^{(enc)} = \{h_1^{(enc)}, \dots, h_n^{(enc)}\}$, where $h_i^{(enc)} \in \mathbb{R}^d$, d is the space dimension, $i \in \{1, ..., n\}$ and $n = |s^t|$. We define the encoder as:

$$h^{(enc)} = encoder(x; \theta^{(enc)}), \tag{6.1}$$

where $\theta^{(enc)}$ are the encoder's trainable parameters.

[2] With the same $[SEP]$ token, the questions and verbalized answers are separated inside the conversation history.

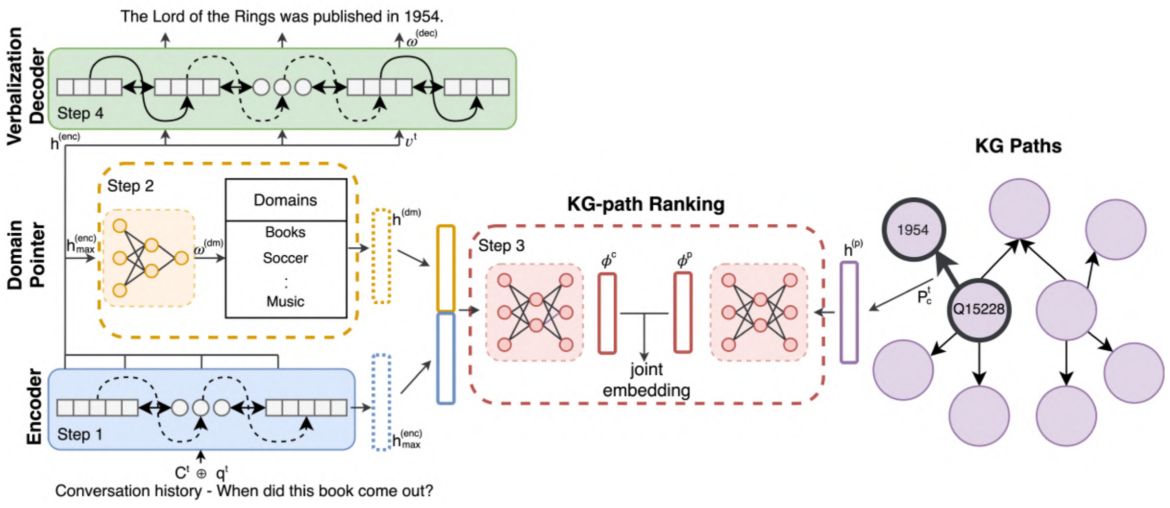

Figure 6.2: PRALINE (**P**ath **R**anking for conversAtionaL questIon aNswEring) architecture. It consists of four modules: 1) A Transformer-based encoder that encodes the input question q^t and conversational history C^t to produce the encoder contextual embeddings $h^{(enc)}$. 2) A domain identification pointer-based network to identify the domain of the input conversation given the contextual embeddings $h^{(enc)}$. 3) A ranking module that learns a joint embedding space ϕ^c, ϕ^p for the conversation (contextual embeddings $h^{(enc)}$ concatenated with selected domain KG embeddings $h^{(dm)}$) and the context path P_c^t. 4) A GPT-based verbalization decoder that generates fluent answer responses while maintaining grammatical correctness.

6.4.2 Domain Identification Pointer

The second step of our framework is a domain identification pointer network. This module is responsible for identifying the KG domain of the input sequence s^t and employs a pointer architecture inspired from [184]. In general, pointer networks are robust to handle different vocabulary sizes for each time step [184] which was our rationale for their integration in PRALINE. Another advantage of using pointer networks compared to simple classifiers is that the vocabulary of the domains can be updated during evaluation or inference.

We define the vocabulary as $V^{(dm)} = \{\tau_1, \ldots, \tau_{n_{dm}}\}$, where n_{dm} is the total number of domains in the KG. To compute the pointer scores for each domain candidate, we use the encoder contextual embeddings $h^{(enc)}$. We model the pointer networks with a feed-forward linear network followed by a softmax layer. We can define the domain pointer as:

$$\omega_i^{(dm)} = softmax(\boldsymbol{W}_1^{(dm)} u_t^{(dm)}), \tag{6.2}$$

where $\omega_i^{(dm)} \in \mathbb{R}^{|V^{(dm)}|}$ is the probability distribution over the domain vocabulary. The weight matrix $\boldsymbol{W}_1^{(dm)} \in \mathbb{R}^{1 \times d_{kg}}$. Also, $u_t^{(dm)}$ is a joint representation that includes the domain embeddings and the contextual embeddings, computed as:

$$u_t^{(dm)} = tanh(\boldsymbol{W}_2^{(dm)} \tau + h^{(enc)}), \tag{6.3}$$

where the weight matrix $\boldsymbol{W}_2^{(dm)} \in \mathbb{R}^{d \times d_{kg}}$. We denote with d_{kg} the dimension used for domain (KG) embeddings.[3] $\tau \in \mathbb{R}^{d_{kg} \times |V^{(dm)}|}$ are the domain embeddings. $tanh$ is the non-linear layer. Here, the contextual embeddings $h^{(enc)}$ is expanded by the size of the domain vocabulary $|V^{(dm)}|$ in order to match the dimensions.

6.4.3 KG-path Ranking

For our approach, we identify context entities $\mathcal{E}_c$ and extract potential candidates for KG paths similar to [17] and is not part of our overall architecture. Hence, as our contribution in this module is to rank these KG paths effectively. We propose a ranking module by employing two identical sequential networks in order to generate joint embeddings for a conversation (input sequence s^t) and a context path p_c^t at turn t. Each sequential network contains two linear layers separated with a $ReLU$ activation function, and appended with a $tanh$ non-linear layer. Here, as input we consider the concatenation of the encoder contextual embeddings $h^{(enc)}$ together with the domain selected from the domain identification pointer (cf. Equation 6.2). In particular we employ BERT [92] embeddings to embed the selected domain and generate the representation $h^{(dm)}$. In this way we incorporate also the domain information when we create the joint embeddings. For a conversation, the encoder contextual embeddings are $h^{(enc)}$ where $h^{(enc)} \in \mathbb{R}^{|s^t| \times d}$. The contextual embeddings contain representations of dimension space d for each token of the input sequence s^t. While, for the domain embedding $h^{(dm)}$ and for each context path embedding $h^{(p)}$ (both initialized using BERT embeddings), we have $h^{(dm)} \in \mathbb{R}^{d_{kg}}$ and $h^{(p)} \in \mathbb{R}^{d_{kg}}$, respectively (implementation details explained in 6.7). In order to match the space dimensions $\mathbb{R}^{|s^t| \times d}$ and $\mathbb{R}^d$, we apply a max pooling layer to the encoder contextual embeddings $h^{(enc)}$ before forwarding it to the sequential network of the module. We define this as:

$$h_{max}^{(enc)} = max_0 h^{(enc)}, \tag{6.4}$$

where max_0 indicates the max operation performed in dimension $zero$, and $h_{max}^{(enc)} \in \mathbb{R}^d$. Overall, we define the module sequential networks as:

$$\begin{aligned} \phi^c &= tanh(\boldsymbol{W}_2^{(crk)} ReLU(\boldsymbol{W}_1^{(crk)} [h_{max}^{(enc)}; h^{(dm)}])), \\ \phi^p &= tanh(\boldsymbol{W}_2^{(prk)} ReLU(\boldsymbol{W}_1^{(prk)} h^{(p)})), \end{aligned} \tag{6.5}$$

where $\boldsymbol{W}_1^{(crk)} \in \mathbb{R}^{d \times 2d}$, $\boldsymbol{W}_1^{(prk)} \in \mathbb{R}^{d \times d}$ are the weight matrices for the first linear layers. $\boldsymbol{W}_2^{(crk)} \in \mathbb{R}^{d \times d}$, $\boldsymbol{W}_2^{(prk)} \in \mathbb{R}^{d \times d}$ are the weight matrices for the second linear layers. $\phi^c \in \mathbb{R}^d$ and $\phi^p \in \mathbb{R}^d$ are the final joint embeddings on space dimension d of the conversation and context path, respectively. During training, PRALINE aims to maximize the cosine similarity between the conversation ϕ^c and gold context paths $\phi^p \in \mathcal{D}^{t+}$ and minimize it for incorrect paths ($\mathcal{D}^{t-}$). On inference, PRALINE jointly embeds the conversation with all context paths and ranks them based on cosine similarity.

6.4.4 Verbalization Decoder

The last module of PRALINE generates the verbalized answer v^t using the encoder contextual embeddings $h^{(enc)}$ of the input sequence s^t. Here, we utilize a BART [93] GPT-based [203] decoder

[3] For our experiments we employ $d_{kg} = d$

for generating the final natural language answer. The decoder vocabulary is defined as:

$$V^{(dec)} = V^{(v)} \cup \{ [ANS] \}, \tag{6.6}$$

where $V^{(v)}$ is the vocabulary with all the distinct tokens from the conversations. We also employ an additional helper token to to specify the position of the answer a^t on the final verbalized sequence. A linear layer and a softmax follow the decoder in order to calculate each token's probability score in the vocabulary. We define the decoder stack output as follows:

$$\begin{aligned} h^{(dec)} &= decoder(h^{(enc)}; \theta^{(dec)}), \\ \omega_i^{(dec)} &= softmax(\boldsymbol{W}^{(dec)} h_i^{(dec)}), \end{aligned} \tag{6.7}$$

where $h_i^{(dec)}$ is the hidden state in time step i, $\theta^{(dec)}$ are the decoder trainable parameters, $\boldsymbol{W}^{(dec)} \in \mathbb{R}^{|V^{(dec)}| \times d}$ are the linear layer weights, and $\omega_i^{(dec)} \in \mathbb{R}^{|V^{(dec)}|}$ is the probability distribution over the decoder vocabulary in time step i. $|V^{(dec)}|$ denotes the decoder's vocabulary size.

6.5 Multi-Task Learning

PRALINE consists of three modules for which a loss function is applied. The encoder is trained based on the signal received from the domain identification pointer, ranking module, and verbalization decoder. For training simultaneously all the modules/tasks, we perform a weighted average of all the single losses as follows:

$$L = \lambda_1 L^{dm} + \lambda_2 L^{rk} + \lambda_3 L^{dec}, \tag{6.8}$$

where $\lambda_1, \lambda_2, \lambda_3$ are the relative weights learned during training to consider the difference in magnitude between losses. L^{dm} and L^{dec} are the respective negative log-likelihood losses of the domain identification pointer and verbalization decoder modules. While L^{rk} is the cosine embedding loss for the ranking module. These losses are defined as:

$$\begin{aligned} L^{dm} &= -\sum_{j=1}^{m} logp(y_j^{(dm)}|x), \\ L^{rk} &= \begin{cases} 1 - cos(\phi^c, \phi^p), & \text{if } y^{(rk)} = 1 \\ max(0, cos(\phi^c, \phi^p) - \alpha), & \text{if } y^{(rk)} = -1 \end{cases}, \\ L^{dec} &= -\sum_{l=1}^{n} logp(y_l^{(dec)}|x), \end{aligned} \tag{6.9}$$

where n is the length of the gold answer verbalization. $y_j^{(dm)} \in V^{(dm)}$ are the gold labels for the domain identification pointer and $y_l^{(dec)} \in V^{(dec)}$ are the gold labels for the verbalization decoder. $y^{(rk)} \in \{1, -1\}$ are the gold labels for the ranking module. $cos(\cdot)$ is the normalized cosine similarity and α is the margin. The model benefits from each module's supervision signals, which improves the performance in the given task (cf. Section 6.8). Algorithm 1 illustrates high-level pseudo-code for PRALINE's learning process.

Algorithmus 1 : Learning Algorithm of PRALINE

```
Input : Training set $S_{train} = \{(q^t, C^t, v^t, \tau^t, \mathcal{D}_c^{t+}, \mathcal{D}_c^{t-})\}$
1  for $S_{batch} \in S_{train}$ do
2      $q_b \leftarrow getQuestions(S_{batch})$
3      $C_b \leftarrow getConvHistory(S_{batch})$
4      $v_b \leftarrow getVerbalizedAnswers(S_{batch})$
5      $\tau_b \leftarrow getDomains(S_{batch})$
6      $\mathcal{D}^+ \leftarrow getPositivePaths(S_{batch})$
7      $\mathcal{D}^- \leftarrow getNegativePaths(S_{batch})$
8      $y_b^{(rk)} : y_b^{(rk)} \in \mathbb{R}^{\{1,-1\} \times b}$
9      for $y_i^{(rk)} \in y_b^{(rk)}$ do
10         if $y_i^{(rk)} = 1$ then
11             $p_i \sim samplePath(\mathcal{D}^+)$
12         else
13             $p_i \sim samplePath(\mathcal{D}^-)$
14         end
15     end
16     begin PRALINE forward
17         $h_b^{(enc)} \leftarrow PRALINE.encoder(q_b, C_b)$
18         $\omega_b^{(dm)} \leftarrow PRALINE.domainPointer(h_b^{(enc)})$
19         $h_b^{(p)} \leftarrow embedPaths(p_b)$
20         $h_b^{(dm)} \leftarrow embedDomains(\tau_b)$
21         $\phi_b^c, \phi_b^p \leftarrow PRALINE.ranking(h_b^{(enc)}, h_b^{(dm)}, h_b^{(p)})$
22         $\omega_b^{(dec)} \leftarrow PRALINE.decoder(h_b^{(enc)}, v_b)$
23     end
24     $L_b^{dm} = \frac{1}{b} \sum_{i=1}^{b} - \sum_{j=1}^{m} logp(y_j^{(dm)} | \omega_i^{(dm)})$
25     $L_b^{rk} = \frac{1}{b} \sum_{i=1}^{b} \begin{cases} 1 - cos(\phi_i^c, \phi_i^p), & \text{if } y_i^{(rk)} = 1 \\ max(0, cos(\phi_i^c, \phi_i^p) - \alpha), & \text{if } y_i^{(rk)} = -1 \end{cases}$
26     $L_b^{dec} = \frac{1}{b} \sum_{i=1}^{b} - \sum_{k=1}^{n} logp(y_l^{(dec)} | \omega_i^{(dec)})$
27     Update PRALINE weights w.r.t. $\lambda_1 L_b^{dm} + \lambda_2 L_b^{rk} + \lambda_3 L_b^{dec}$
28 end
```

6.6 Benchmark with Answer Verbalization

The existing ConvQA datasets [16, 17, 117] lack verbalized answers (cf. Table 6.2). Hence, we extended ConvQuestions [16] and ConvRef [17] dataset with answer verbalization by employing a generation procedure similar to Chapter 3. We describe the process below [204].

Dataset	**Large-scale (>=10k)**	**Complex Ques.**	**Reformulated Ques.**	**Verbalized Ans.**	**Reformulated Ans.**
CSQA [117]	✓	✓	✗	✗	✗
ConvQuestion [16]	✓	✓	✗	✗	✗
ConvRef [17]	✓	✓	✓	✗	✗
"*Verbal*"-ConvQuestions (**ours**)	✓	✓	✓	✓	✗
"*Verbal*"-ConvRef (**ours**)	✓	✓	✓	✓	✓

Table 6.2: Comparison of "*Verbal*"-ConvQuestions and "*Verbal*"-ConvRef with existing conversational KGQA datasets in different dimensions. The lack of answer verbalization and reformulated utterances remains a key gap in the literature.

Data-Generation and Augmentation

We inherit questions from ConvQuestions [16] and ConvRef [17] datasets, which are high quality and large-scale benchmark for conversational QA over Wikidata KG [2]. The datasets were compiled from inputs of crowd workers with conversations from domains such as music, soccer, TV series, books, and movies. The questions incorporate challenging phenomena such as aggregations, compositionality, temporal reasoning, and comparisons.

Initial Answer Verbalization

The first step is to generate the initial answer verbalization from the seed answers given in the original datasets. We group all similar questions or question templates and reword them using a rule-based approach. To maintain consistency across all answers, we cover the question and answer entities using the general tokens ENT, and ANS. We substitute the tokens back to the original position after the first version is generated. Similarly to other works [192], we use box brackets to distinguish the seed answer from the remaining sentence; this is helpful when experimenting with the verbalized answers. For example, for the question "What countries did the main character travel in the book Eat, Pray, Love?" the step would provide an initial verbalized answer: "The main character travels in the book Eat, Pray, Love to [Italy, India, and Indonesia]." Here, the answer is mainly a paraphrased version of the question that includes the seed answers.

Verbalized Answer Reformulations

Once we generate the first verbalized answer, inspired by [26, 138, 139], we employ back translation to produce multiple reformulated instances for the verbalized answer. In particular, as in Chapter 3, the back translation is performed using a Transformer-based model [82] as translators. We produce reformulations using translators for two languages (English-Russian, English-German). For the selection, we considered the models' performance on the WMT'18 dataset [205], including translations between different languages.

Result Validation via Human Annotators

Finally, to ensure the grammatical correctness of all the generated answers, we include two rounds of a peer-review process. Similar to [192], the first round included a set of in-house workers where we asked them to assert the produced results and rephrase them if needed. This step ensures more natural and fluent dialogues. Next, another set of in-house workers were asked to validate the previous step and rephrase if needed. In particular, for the initial verbalization, over 55% required some human interventions for ConvQuestions and 47% for ConvRef. While for reformulated utterances, over 45% required corrections during the peer-reviewed process for both datasets.

Generated Dataset Examples

Recent efforts introduce answer verbalization in a QA dataset [25, 26]. The associated empirical results indicate the effectiveness of answer verbalization [26]. Albeit effective, answer verbalization for ConvQA is an open research direction due to unavailability of dataset(s) (cf. Table 6.2). Here, we manually examine some examples from the ConvQA datasets we extended with verbalized answers and discuss the intentions behind their construction.

Domain	**Conversations**
Soccer	q^1: What was the birth city of Lionel Messi? v^1: The birth city of Lionel Messi was Rosario, Santa Fe. q^2: Is he a member of the Colombian National soccer team? v^2: No, Lionel Messi does not represent Colombia at the international level. q^3: Which national team is he a member of? v^3: He is an Argentina national team player. q^4: Which of their goalkeepers is youngest? v^4: Juan Musso is the youngest goalkeeper. q^5: When did he join the team? v^5: He joined the team in 2019.
TV Series	q^1: What network was Dexter on? v^1: Dexter aired on Showtime. q^2: What year did the show debut? v^2: The show debuted in 2007. q^3: Who starred in it? v^3: Michael C. Hall was the show's star. q^4: And how many seasons did the show last? v^4: Dexter is an 8 season television series. q^5: What was the main location of it? v^5: The show was set in Miami.

Table 6.3: Conversation examples from ConvQuestions dataset extended with verbalized answers. Each conversation in the dataset consists of five turns.

Regularly, all the required information is provided with the question/query for the question answering task. Therefore, to generate verbalized answers in such scenarios, we have to concentrate only on the question context, considering that the seed answer is given. However, in conversational question answering, we have scenarios such as anaphora and ellipsis [16, 117], where the context from previous turns has to be incorporated in order to answer the given question. Hence, we had to consider all these cases when extending the dataset. Table 6.3 illustrates such examples from the ConvQuestions dataset, where conversational context was incorporated to generate the verbalized answers. For instance, in the first conversational example, the user asks the question "*What was the birth city of Lionel Messi?*"; the dataset here includes the verbalized answer "*The birth city of Lionel Messi was Rosario, Santa Fe.*". In the next turn, we have the question "*Is he a member of the Colombian National soccer team?*" where "*he*" refers to "*Lionel Messi*". For such examples, we either provide verbalized answers using the relevant pronoun (e.g "*No, he is not a member of the Colombian National soccer team.*") or even the entity itself, (e.g. "*No, Lionel Messi does not represent Colombia at the international level.*"). Similarly, on the next conversation, which belongs to the "*TV Series*" domain, we provide such answers. More precisely, on turn four, the user asks the question "*And how many seasons did the show last?*" referring to the seed entity "*Dexter*" from first turn. Even in such scenarios, we construct answers that also contain the entity (e.g. "*Dexter is an 8 season television series.*").

6.7 Experimental Setup

In this section, we provide the setup for the experiments. We further describe the resources and metrics we employ.

Model Configurations

For all the modules in the PRALINE framework, we employ a space dimension $d = 768$. We apply residual dropout in different parts and modules of our framework (such as domain pointer and ranking) with a rate of 0.1. For the domains, we initialized the domain KG embeddings using sentence embeddings that implicitly use underlying hidden states from BERT network [92]. We treat each domain as a sentence and feed that as an input to BERT. We receive as output the domain KG representation with a dimension $d_{kg} = 768$. This representation is used for the domain identification task and also for the ranking task. Similar representations have been created for context path candidates. For training, we employ a batch size of 32, a learning rate of $1e-4$, and we train for 120 epochs and store the models' checkpoints. For the optimization, we use the AdamW algorithm with weight decay fix as introduced in [206]. We restrict verbalized answers v^t length to 50 tokens, while PRALINE's input sequence $(C^t + q^t)$ is restricted to 150 tokens. For the domain pointer and verbalization decoder cross-entropy loss, we apply relative weights λ_1 and λ_2 of 0.25. Finally, for the KG-path ranking cosine embedding loss, we use a margin α of 0.1 and relative weight λ_3 of 1.0.

Models for Comparison

For ConvQA over KGs, we compare our framework with two baselines that have been evaluated on both ConvQuestions and ConvRef datasets. The first baseline is CONVEX [16] which detects answers to conversational utterances over KGs in a two-stage process based on judicious graph expansion. First, it detects frontier nodes that define the context at a given turn. Then, it finds high-scoring candidate answers in the vicinity of the frontier nodes. The second baseline and current state-of-the-art is CONQUER [17], an RL-based method for conversational QA over KGs, which leverages implicit negative feedback when users reformulate previously failed questions. A recently proposed model OAT [181] reports values on ConvQuestions that proposes a semantic parsing-based approach. Again, we inherit baseline values from original publications. Furthermore, we cannot compare the answer verbalization performance against these baselines since they do not generate verbalized natural language answers. Hence we compare against existing sequence to sequence architectures such as RNN [71], convolutional [140] and Transformer model [82] to illustrate verbalization task accuracy during datasets extension.

Evaluation Metrics

For evaluating the ConvQA performance, we use the following ranking metrics which are also employed by the previous baselines: 1) Precision at the top rank (P@1) 2) Mean Reciprocal Rank (MRR) is the average across the reciprocal of the rank at which the first context path was retrieved. 3) Hits at 5 (H@5) is the fraction of times a correct answer was retrieved within the top-5 positions. We report precision, recall, and F1-score for the domain identification task, while for answer verbalization, we employ BLEU-4 and METEOR.

6.8 Results

We conduct our experiments and analysis to evaluate the efficacy of PRALINE in ranking the KG paths for extracting answers to the questions asked during a conversation. For achieving that, we want to assess the effect of conversational context on the efficiency of PRALINE and the task-specific (domain identification, verbalization, etc.) performance in the PRALINE framework.

Dataset	**ConvQues.**			**ConvRef**		
Model	P@1	H@5	MRR	P@1	H@5	MRR
CONVEX [16]	0.184	0.219	0.200	0.225	0.257	0.241
CONQUER [17]	0.263	0.343	0.298	**0.358**	0.439	0.395
OAT [181]	0.250	-	0.260	-	-	-
PRALINE	**0.292**	**0.529**	**0.398**	0.335	**0.599**	**0.441**

Table 6.4: Overall results on employed datasets. The effect of incorporating conversational context in PRALINE has positively impacted empirical results, achieving better results than baselines. Best values are in bold.

Overall Performance on ConvQA datasets

Table 6.4 summarizes the results comparing PRALINE against the previous baselines. PRALINE outperforms previous baselines in all metrics on the ConvQuestions dataset. Specifically, for P@1, PRALINE performs by 0.029 points better than CONQUER, 0.108 points compared to CONVEX, and 0.042 points against OAT. For H@5 and MRR, the margin is even more prominent, with 0.186 and 0.100 total points, respectively, compared against CONQUER. While for CONVEX, the margins

Dataset	**ConvQuestions**									
Domain	**Movies**		**TV Series**		**Music**		**Books**		**Soccer**	
Models	H@5	MRR	H@5	MRR	H@5	MRR	H@5	MRR	H@5	MRR
CONVEX [16]	0.355	0.305	0.269	0.218	0.293	0.237	0.303	0.246	0.284	0.234
CONQUER [17]	0.357	0.316	0.382	0.325	0.320	0.263	0.464	0.417	0.310	0.268
PRALINE	**0.561**	**0.426**	**0.457**	**0.378**	**0.405**	**0.279**	**0.739**	**0.599**	**0.492**	**0.344**
Dataset	**ConvRef**									
Domain	**Movies**		**TV Series**		**Music**		**Books**		**Soccer**	
Models	H@5	MRR	H@5	MRR	H@5	MRR	H@5	MRR	H@5	MRR
CONQUER [17]	0.436	0.405	0.442	0.392	0.398	**0.357**	0.554	0.502	0.360	0.316
PRALINE	**0.567**	**0.429**	**0.545**	**0.466**	**0.495**	0.329	**0.835**	**0.659**	**0.564**	**0.378**

Table 6.5: To compare the KG path ranking performance, we report fine-grained results across different domains of both benchmarks on ranking metrics. CONVEX does not report domain-specific values on ConvRef dataset, hence omitted from the respective table. PRALINE maintains an empirical edge on baselines while ranking the KG paths. Best values are in bold.

increase to 0.310 for H@5 and 0.198 for MRR. For OAT, only the MRR is available, and PRALINE outperforms it by 0.138. For the ConvRef dataset, PRALINE performs better in all metrics compared to CONVEX, where the margin for all metrics is more than 0.100 absolute points. Moreover, it surpasses CONQUER on ranking metrics H@5 and MRR with 0.160 and 0.046 points. ConvRef is an extended version of ConvQuestions with multiple question reformulations. CONQUER was sophisticatedly designed to leverage those reformulations and boost the results [17]. On the other hand, PRALINE treats the reformulated questions in the same manner as it does with the original questions from the ConvQuestions benchmark. Therefore the increase for P@1 is not that significant. The effect of reformulated questions as the additional context has not been extensively studied in the scope of the work, and we leave it for future work.

Ranking Performance Across Domains

We further investigate the ranking performance of PRALINE across different domains for both benchmarks considering this is the main focus of our work. Table 6.5 illustrates detailed ranking results for H@5 and MRR as both are ranking metrics. As shown, for the ConvQuestions benchmark, PRALINE superiority is evident in all five domains against the baselines. However, we obtain the lowest ranking results in the *Music* domain with 0.405 for H@5 and 0.279 for MRR. Analyzing some of the conversational examples in that domain indicated that we did not have gold positive KG paths for all the instances in the dataset. Therefore, PRALINE could not have a complete training process for all possible conversations and paths. Such issues have also impacted the baselines. Next, we can see that the highest-ranking results are obtained in the *Books* domain, where PRALINE achieves the impressive 0.739 for H@5, which is almost 0.300 points higher than CONQUER. Furthermore, the highest MRR (0.599) is achieved in this domain. The results in these two domain are heavily impacted because we had gold-standard paths for most dataset instances. Also, the number of KG

	ConvQuestions				ConvRef			
	H@5	H@10	MR	MRR	H@5	H@10	MR	MRR
Movies	0.561	0.578	21.04	0.426	0.567	0.625	23.77	0.429
TV Series	0.457	0.621	54.89	0.378	0.545	0.659	50.10	0.466
Music	0.405	0.467	124.34	0.279	0.495	0.566	96.19	0.329
Books	0.739	0.815	7.00	0.599	0.835	0.871	4.99	0.659
Soccer	0.492	0.546	126.92	0.344	0.564	0.596	115.76	0.378

Table 6.6: PRALINE detailed ranking results across different domains. We additionally report results for Hits@10 and Mean Rank (MR). For MR lower is better.

relations used in positive paths is proportionately smaller than other domains, positively impacting the ranking task for all models. On the ConvRef benchmark, PRALINE still outperforms CONQUER on all domains. Here, PRALINE shows substantially improved results in two domains (*TV series*, *Books*) compared to results in ConvQuestions. We conclude that incorporating the entire conversational history extended with additional context (verbalized answers and domain information) for the ranking task has a positive impact on overall empirical performance of PRALINE.

Table 6.6 presents PRALINE's detailed ranking results for Hits@5, Hits@10, Mean Rank (MR) and Mean Reciprocal Rank (MRR) ranking metrics. We achieve the highest scores in the "*Books*" domain, where we also have the lowest MR with only 4.99 on the ConvRef benchmark. On the other hand, we have the lowest scores in "*Music*" and "*Soccer*" domains.

6.9 Ablation Study

We perform various ablation studies on PRALINE to illustrate the effectiveness of the proposed approach and related architecture choices. Table 6.7 summarizes the results of the ablation studies.

Effect of Verbalized Answers

To show the effectiveness of using verbalized answers in conversational history, we perform an ablation experiment where we remove and replace them with non-verbalized answers extracted from the KG (e.g. entities, literals). We can observe that the ranking performance drops significantly. In particular, we obtain a drop between 0.060 – 0.080 for H@5 and MRR ranking metrics and 0.027 for P@1. Verbalized answers provide additional context in the conversational history and, therefore, support PRALINE to create more accurate representations for the ranking task. Such context is crucial for our results.

Dataset	ConvQues.			ConvRef		
Model	P@1	H@5	MRR	P@1	H@5	MRR
PRALINE	0.292	0.529	0.398	0.335	0.599	0.441
w/o verb. ans.	0.265	0.441	0.324	0.279	0.503	0.397
w/o domain	0.247	0.436	0.296	0.266	0.472	0.356
w/o full conv.	0.214	0.375	0.299	0.247	0.449	0.324
train separately	0.255	0.413	0.328	0.304	0.529	0.408

Table 6.7: The effectiveness of including verbalized answers, entire dialog history, and domain information. The first row (from top) contains the results of PRALINE with all available contexts. The second and third-row selectively remove the verbalized answers and domain information respectively. 4th row omits the full conversational history and includes only the previous turn. In the last row, we show results when we train modules independently, illustrating the advantage of joint training of PRALINE modules.

Effect of Domain Information

For the second ablation experiment, we remove the domain information (domain pointer module) from PRALINE. We can see the importance of such information in our approach. All metrics results have dropped, indicating the effect of it. In PRALINE, domain information is used to improve the conversation representation and indirectly filter KG paths that are not relevant. Usually, such paths contain KG relations that are not used in the particular domain and add noise while ranking the correct paths.

Effect of Full Conversational History

We study the empirical advantage of incorporating the entire dialog history. Hence, we configured PRALINE to consider dialog history from the previous turn, disregarding full history (w/o full conv). As a result, there is a drop in the performance for PRALINE (w/o full conv.), illustrated in Table 6.7. Consequently, we conclude that conversational contexts (full dialog history with answer verbalization and domain information) positively impact the KG path ranking.

Effect of Joint Training

As a last ablation experiment, we study the effectiveness of jointly training PRALINE modules. We do that by training each module independently without sharing any sub-module information (e.g., encoder). Ablation results indicate lower scores for all metrics. Furthermore, we observed that all modules were overfitting much faster during this experiment than PRALINE's joint training. The observed behavior is reasonable since the more correlated tasks we are jointly learning, PRALINE generates better embedding representations that capture all the tasks, yielding a lower chance of overfitting.

6.10 Task Analysis

We calculated the task-specific performance of different modules in our framework to justify choosing various modules. Table 6.8 illustrates performance of domain identification pointer on both benchmarks. The robust results justify the use of pointer network, which also complement the ablation study in Table 6.7 for PRALINE (w/o domain) configuration.

Table 6.9 presents the results of PRALINE's verbalization decoder. We obtain a BLEU-4 score of 0.289 on ConvQuestions and 0.327 on ConvRef, while for METEOR, the scores are 0.626 and 0.684, respectively. Here, the score margins between the benchmarks are larger than the domain pointer task. The additional questions in ConvRef further support PRALINE to avoid overfitting for the answer verbalization task. We conclude that the scores are considered appropriate and support PRALINE's performance as reported in Tables 6.7 and 6.4.

Table 6.10 summarizes the results for the answer verbalization task and compares them with other sequence to sequence baseline models. We perceive that PRALINE outperforms the other baselines on all five domains for both BLEU-4 and METEOR metrics. Interestingly, all models perform poorly in the "*Movies*" and "*Soccer*" domains. This occurs due to similar KG relations (e.g. "*plays for*") they share with other domains (e.g. "*Movies*", "*Soccer*" and "*Books*"). Furthermore, all models perform relatively well on domains such as "*Music*" and "*Books*."

Dataset	**ConvQuestions**			**ConvRef**		
Task	Pres.	Rec.	F1	Pres.	Rec.	F1
Domain Pointer	0.951	0.946	0.947	0.958	0.959	0.952

Table 6.8: Domain identification results.

Dataset	**ConvQuestions**		**ConvRef**	
Task	BLEU-4	METEOR	BLEU-4	METEOR
Verbaliz. Decoder	0.289	0.626	0.327	0.684

Table 6.9: Answer verbalization results.

Domain	**Movies**		**TV Series**		**Music**		**Books**		**Soccer**	
Models	BLEU-4	METEOR	BLEU-4	METEOR	BLEU-4	METEOR	BLEU-4	METEOR	BLEU-4	METEOR
RNNSeq2seq [71]	0.125	0.441	0.108	0.463	0.206	0.499	0.149	0.391	0.103	0.456
CNNSeq2seq [140]	0.117	0.453	0.115	0.459	0.238	0.519	0.133	0.330	0.055	0.404
Transformer [82]	0.263	0.549	0.230	0.511	0.359	0.639	0.375	0.626	0.199	0.570
PRALINE	**0.297**	**0.585**	**0.271**	**0.625**	**0.386**	**0.667**	**0.404**	**0.693**	**0.236**	**0.608**

Table 6.10: Detailed answer verbalization results on "*Verbal*"-ConvQuestions dataset across different domains.

6.11 Error Analysis

ConvQA

For the error analysis, we randomly sampled 250 incorrect predictions (with equal predictions from each domain). We detail the reasons for two types of errors observed in the analysis:

Incorrect Ranking of Paths with Semantically Similar KG Relations. PRALINE often wrongly-ranks paths when they contain semantically similar relations. For instance, given the question "*What kind of book is it?*" and its entire conversation history: q^1) "*What is the name of the writer of The Secret Garden?*" v^1) "*The name of the writer of The Secret Garden is Frances Eliza Hodgson Burnett.*" q^2) "*Where does the story take place?*" v^2) "*The story takes place in Yorkshire.*" q^3) "*When was the book published?*" v^3) "*The book was published in 1910.*". PRALINE is required to find the gold path that contains the KG relation "main subject (P921)" since this one points to the correct answer, which is "*adventure (Q1436734)*". However, PRALINE here ranks higher paths that contain the KG relation "*genre (P136)*". For the example mentioned above, there are three KG paths with relation "*genre*", and all of them are ranked in the top three positions. As we can see, the relations "*main subject*" and "*genre*" are semantically similar and, therefore, hard to distinguish which one to rank higher. For the particular example, the relation "*genre*" is ranked higher since it is used more across the gold KG paths in training data. In this work, we focused on context derived from the conversation and have not considered widely available KG context such as entity/relation aliases, types, etc.

Absence of Gold KG Paths. Several examples (over 25%) with missing gold paths in training datasets significantly affect the learning process. For the test sets, there were 19% of conversational turns without gold KG paths. These examples are directly counted as wrong instances and negatively affect our results. With a more sophisticated annotation process for gold KG paths, PRALINE results would have been improved.

Answer Verbalization

We randomly sampled 200 incorrect verbalized answers generated from PRALINE. We detail the error rates for three categories and compare them with those from sequence to sequence models. The first one, "*Grammatical*", refers to examples where the verbalized answer can be understood but contain some grammatical errors such as a mismatch between the noun and verb forms. Next, the "*Semantic*" error class refers to the cases where the primary meaning of the answer has changed; this can occur by introducing new content (e.g., entities) or by omitting essential parts of the content. The last error

Models	Grammatical	Semantic	Entity Related
RNNSeq2seq	85%	88%	79%
CNNSeq2seq	86%	91%	77%
Transformer	41%	48%	35%
PRALINE	18%	29%	24%

Table 6.11: Error rates of PRALINE answer verbalization compared to other baseline models.

class is the "*Entity Related*", which refers to generated answers where PRALINE failed to copy the correct entities from the input utterance or replace them with pronouns. Table 6.11 presents the error rates for the three categories. We can observe that PRALINE contains the lowest error rates and, therefore, validates its superior performance. Interestingly, PRALINE's error rates are lower than 30% for all classes. In particular, for the first class ("*Grammatical*"), the error rate is 18%, indicating that grammatical errors are rare in the generated answers. For the second class ("*Semantic*"), we see the error rate increasing to 29%, which usually happens when the results do not focus only on the primary meaning of the question but rather on less related details from the conversational history. For the last class ("*Entity Related*"), the rate is 24%; here, errors are observed when the verbalized answer contains other entities from the conversational history. Such errors occur when the question refers to an entity in conversational history, and the model fails to determine which one. Our empirical study provides a glance at the different types of errors that may arise while targeting the tasks of ConvQA over KGs with verbalized answers.

6.12 Summary

In this chapter, our objective was to study if conversational context (entire dialog history with verbalized answers) positively impacts the ranking of KG paths for retrieving answers to a question. The multi-task learning approach implemented and its associated empirical advantage over baselines established the necessity of incorporating such context for the ConvQA task. We also conclude that a joint embedding of conversation and KG-paths in a homogeneous space positively impacts the overall ranking metrics. Furthermore, our systematic ablation studies illustrate each conversation context's impact (entire conversation history, verbalized answer, and domain information) on PRALINE's performance. It also justifies our rationale to extend existing ConvQA datasets with verbalized answers.

CHAPTER 7

Conclusion and Future Directions

In this thesis, we follow the scientific method for improving the state-of-the-art of Conversational Question Answering over Knowledge Graphs and Answer Verbalization via the Multi-Task Learning paradigm. First, we defined the research problem in Chapter 1 and discussed the significant challenges that we require to overcome to achieve the research objective. Chapter 2 discussed all the fundamentals and necessary background concepts needed for this thesis. We have broken down the main research problem into four questions to achieve our research objective and overcome the challenges. We tackled these research questions in the subsequent four chapters of the thesis.

The following sections summarize our contributions, elaborating the main findings that validate our research questions.

7.1 Review of the Contributions

This section summarizes the contributions provided in the thesis from Chapters 3, 4, 5, and 6. With this thesis, we advance the fields of conversational question answering over knowledge graphs and answer verbalization by providing novel approaches and resources that address the main challenges. In this respect, our contributions answer four research questions which we review below. We state the research question and then the contributions towards them.

First, we investigated the impact of multiple diverse responses in the answer verbalization task.

RQ1: How do multiple paraphrased answers affect the performance of answer verbalization?

Answer verbalization plays a vital role in providing fluent responses to the users. Currently, there is a lack of KGQA datasets that support answer verbalization. Existing datasets [24, 25] only provide one verbalized response per question. However, an answer can be phrased in various ways while maintaining the same semantic meaning. Multiple paraphrased responses can help conversations become more flexible and intuitive. Motivated by the idea that not all humans interpret an answer the same way, while all interpretations retain the same meaning, we developed a resource with multiple paraphrased answers for each question. However, our main objective was to study whether machine learning models' performance can be positively affected by them. Since intuitively supplying multiple gold samples to a model instead of one can have a positive impact. Our empirical analysis indicated higher scores for generation metrics when employed multiple paraphrased answers compared to one or even two. We obtained the same in three different deep neural approaches, each tested with two different inputs (question or SPARQL queries).

Contributions to Research Question *RQ1* are summarized as follows:

- We provide a semi-automated framework for generating multiple paraphrase responses for each question using techniques such as back-translation.
- We present the first question answering dataset with multiple paraphrased responses. In particular, the dataset consists of up to eight unique paraphrased responses for each dataset question that can be answered using DBpedia as the underlying KG.
- We provide evaluation baselines that serve to determine our dataset's quality and define a benchmark for future research.
- We analyze the performance of various models when trained with one or more paraphrased answers.

Second, we explore integrating logical forms as an additional context for improving answer verbalization performance via multi-task learning.

RQ2: How can we incorporate logical forms to improve the performance of answer verbalization via multi-task learning?

The goal of the answer verbalization task is to produce fluent, natural language responses to a user question. Existing baselines utilize only the question as a source of information for generating verbalized responses. Yet, a question can be reformulated in numerous ways that might affect the underlying model's learning process since there are no constant patterns to generate the verbalized answers. Hence, an additional context with more prominent patterns will enable the model to determine similar questions and satisfy them with similar responses. Our objective here is to improve answer verbalization performance further. On the experiments we had for the previous research question (*RQ1*), we noticed that providing the queries instead of the question as input to the models delivers better results due to the constant input pattern templates that the queries have. Usually, with questions, we have a different reworded version for the same template. In an attempt to leverage that, we proposed to employ logical form queries as an additional context alongside the questions. In this way, the model can learn from both sources and generate better verbalizations. We supplemented this with multi-task learning, where our approach will learn multiple sub-tasks jointly for verbalizing answers. Our results indicated that combining logical form and question information performs better compared to utilizing one at a time.

Contributions to Research Question *RQ2* are summarized as follows:

- We introduce a multi-task-based hybrid answer verbalization framework that consists of four modules trained simultaneously.
- We propose a similarity threshold and cross attention modules to determine the relevance between the inputs and fuse information to employ a hybrid strategy.

- We provide an extensive evaluation and ablation study of the proposed framework on three QA datasets with answer verbalization. Our evaluation results establish a new baseline for the answer verbalization task, which we believe will drive future research in a newly studied problem.

Third, we intend to improve conversational question answering over knowledge graphs via multi-task learning.

RQ3: How can we develop better and more efficient multi-task semantic parsing approaches for conversational question answering?

One of the main methodologies for approaching KGQA is semantic parsing, which requires a grammar of actions to build executable queries over the KG. These actions can vary based on the complexity and type of questions. While for KGQA, there exist numerous grammars, they cannot be directly applied to the ConvQA task since they do not address conversational scenarios (e.g., clarification questions). Furthermore, incorporating KG information such as entities, relations, and types is crucial for particular conversational scenarios and the ConvQA task in general. Therefore, one of the open challenges is discovering methods that specify the relevant information in the KG and incorporate it. Here, we targeted the task of conversational (complex) question answering over a large-scale knowledge graph. We reviewed existing related works in the domain and proposed two approaches that employ state-of-the-art deep neural architectures. In particular, we identified possible drawbacks or space for improvement. We developed new novel multi-task neural semantic parsing approaches that surpass the baselines in most question types and overall. We found the necessity to introduce a new semantic grammar for ConvQA and proposed solutions that can better determine the KG information required. Our experiments validate the proposed architectural choices and all our assumptions.

Contributions to Research Question *RQ3* are summarized as follows:

- We introduce two multi-task learning frameworks for conversational question answering over large scale knowledge graph.
- We propose an architecture of stacked pointer networks for incorporating knowledge graph information on conversational question answering task.
- We employ a Transformer model supplemented with a Graph Attention Network to exploit the correlations between (entity) types and predicates due to its message-passing ability between the nodes.
- We propose a novel entity recognition module that detects, links, filters, and permutes all relevant entities.
- We propose a reusable grammar for neural semantic parsing to define various logical forms that can be executed on a KG for fetching answers to conversational questions.
- We empirically study the proposed architectural design choices through an extensive evaluation, ablation study, and multiple analyses.

Fourth and last, we comprise answer verbalization into conversational question answering to improve the performance.

> ***RQ4***: How can answer verbalization be leveraged to improve the performance of conversational question answering?

Conversational context plays a vital role in improving ConvQA performance. In particular, we require the entire dialog history to answer conversational questions. Therefore, available contexts have to be identified and effectively incorporated to improve performance. Consequently, verbalized answers can provide context that might be helpful for particular conversational scenarios. However, it is challenging to integrate the answer verbalization task into the ConvQA task. Since in a multi-task learning framework, we must determine how to establish a connection between the sub-tasks for effectively sharing training signals and controlling the learning process of each sub-task for optimal performance. With the previous research questions, we individually targeted the tasks of ConvQA and answer verbalization while utilizing multi-task learning. Here, we wanted to observe whether answer verbalization textual context can support the ConvQA task for improving the performance. We persisted with the multi-task learning paradigm and proposed an approach that utilizes the entire conversational history with verbalized answers when seeking to rank KG paths for identifying the answer to a given conversational question. Our systematic experiments indicated that answer verbalization and multi-task learning positively impact the ranking task, and therefore we obtained improved scores compared to baselines.

Contributions to Research Question *RQ4* are summarized as follows:

- We propose a ConvQA over KGs approach that jointly models the available conversational context (full dialog history with verbalized answers) and KG paths in a common space for learning joint embedding representations to improve KG path ranking.
- We extend the existing ConvQA datasets with verbalized answers.
- We systematically study the impact of incorporating additional context on the ConvQA performance. Results on standard datasets show a considerable improvement over previous baselines. Our evaluation results establish a new baseline, which we believe will drive future research in a new way of developing such frameworks.

7.2 Limitations and Future Directions

This thesis presented our findings for advancing the fields of conversational question answering over knowledge graph and answer verbalization with the power exposed by deep learning algorithms and the multi-task learning paradigm. Although we achieved adequate results to validate our research questions, these contributions are meant to ignite a new research conversation. Likewise, a few limitations of this research have not been covered in the scope of the thesis. We list the following:

- Our work on the answer verbalization task focuses on improving the overall performance. The presented methods aim to provide improved performance via multiple paraphrased responses (Chapter 3) and additional context to support the model learning process (Chapter 4). It would be interesting to investigate any possible bias in the task. For instance, to study whether the model favors generating particular responses and how they are provoked. We can connect this with the explainability part of the model, where we aim to understand how and why it makes certain decisions.

- The main pitfall of the semantic parsing-based ConvQA approaches (Chapter 5) is the error propagated from the annotation and preprocessing step since we require the gold logical forms to train the models in a supervised way. The models focus on learning the correct logical forms during training, but there is no feedback signal from the resulting answer from the KG. Reinforcement learning might be a way to address this. Furthermore, improving the performance of challenging question types (e.g., "*Clarification*") remains an open question.
- For the ranking-based ConvQA task (Chapter 6), heavy preprocessing is generally required for the approach since we need to identify the context entities and candidate KG paths for each conversational turn and embed them in order to have a quicker and more efficient training process. The preprocessing might not be an option in a real-world application, and without it, concerns are raised regarding the overall run time of the model. We believe that the whole process can be optimized and performed relatively quickly; however, it was not our focus to perform such experiments.

The contributions in this thesis pave the way for a more extensive research agenda that will foster further research. A few of such directions are enumerated below:

Answer Verbalization:

- Currently, the task heavily relies on the user query and information produced by the underlying model (e.g., logical forms). Recently, knowledge graphs have been used as a source of external knowledge in tasks such as entity and relation linking, which are also prominent for question answering. It is yet to be studied if external knowledge from KGs may positively impact the answer verbalization.
- There are empirical evaluations that for AI systems, the explanations regarding the retrieved answers improve trustworthiness, especially in wrong prediction [207, 208]. Hence, how an answer verbalization can be explained remains an important open research direction.
- Our work focused on English as the underlying language, partly because of the availability of datasets. Despite the contributions made, the field still lacks large-scale resources, especially multilingual ones. It would even be more plausible if the new resources were built using qualifier information from KGs to provide informative answer verbalization.

Conversational Question Answering over Knowledge Graphs:

- The proposed semantic parsing systems perform exceptionally well on an open domain large-scale dataset for conversational question answering. The systems answer complex question types such as "*Comparative*" and "*Quantitative*" which yet cannot be handled with other non-KGs approaches (e.g., Machine Reading Comprehension), this further encourages the use of KGs for the QA task. Nevertheless, these systems are restrictive to the question types shown during training, and we do not know how they can reason unseen question types on inference. Inductive learning might be a solution and, therefore, a possible direction for the task.

- Path ranking is a relatively new method for approaching the task. While it is more difficult to comprehend the predictions compared to semantic parsing models, path ranking will be favored since it does not depend on any annotation process or data. We see an emergence of future ConvQA systems towards that direction; however, the explainability of ranking strategies remains a tough challenge and is eventually to be addressed.

- We presented the first ConvQA approach that performs answer verbalization for improving performance. We envision this as the future for those systems. Answer verbalization is essential for expressing and supplying better and more natural responses and can be helpful by providing additional context for particular conversational scenarios. ConvQA systems can further leverage answer verbalization to establish interactive communication and employ possible user feedback to clarify ambiguous requests. In particular, Interactive ConvQA is a beneficial research direction for further improving these systems, and we see answer verbalization as a critical aspect of them.

Multi-Task Learning:

- For the approaches presented in this thesis, we employ *multi-task learning* via *hard parameter sharing* as the default learning paradigm where a Transformer encoder is simultaneously trained to generate representations that are shared across all the tasks. Multi-task learning can be crucial for approaching complicated tasks such as question answering. We expect the development of massive and flexible multi-task learners able to perform numerous tasks. Question answering will be achieved by combining parts of them, and for any new knowledge learned, the model will update a comparatively small number of parameters. Likewise, we believe that multi-task learning is essential in developing artificial intelligence with more human-like qualities.

In summary, research on knowledge graph question answering is continuously evolving. Fields such as conversational question answering and answer verbalization are getting more attention and will keep evolving in the future. During this thesis, we advanced the state of the art in conversational question answering and answer verbalization on several facades by setting up benchmarks and developing approaches dedicated for them. More precisely, we proposed: *i)* an efficient and scalable approach to extend a KGQA dataset with multiple paraphrased answers, *ii)* a multi-task learning framework dedicated to verbalizing answers given multiple sources, *iii)* multi-task neural semantic parsing approaches for ConvQA, and *iv)* an approach that leverages answer verbalization for improving ConvQA task via KG path ranking.

Future research work can build upon the resources and approaches developed as contributions presented during this thesis. These contributions could provide a foundation for future conversational AI systems.

Bibliography

[1] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, et al., *Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia*, Semantic web **6** (2015) 167 (cit. on pp. 1, 9, 62).

[2] D. Vrandečić and M. Krötzsch, *Wikidata: a free collaborative knowledgebase*, Communications of the ACM **57** (2014) 78 (cit. on pp. 1, 62, 99).

[3] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: a core of semantic knowledge," *Proceedings of the 16th international conference on World Wide Web*, 2007 697 (cit. on p. 1).

[4] D. Diefenbach, V. Lopez, K. Singh, and P. Maret, *Core techniques of question answering systems over knowledge bases: a survey*, Knowledge and Information systems **55** (2018) 529 (cit. on pp. 1, 92).

[5] Y. Lan, G. He, J. Jiang, J. Jiang, W. X. Zhao, and J.-R. Wen, "A Survey on Complex Knowledge Base Question Answering: Methods, Challenges and Solutions," *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, Survey Track, International Joint Conferences on Artificial Intelligence Organization, 2021 4483 (cit. on pp. 1, 90).

[6] H. Zafartavanaelmi, *Semantic Question Answering Over Knowledge Graphs: Pitfalls and Pearls*, (2021) (cit. on pp. 1, 2, 44).

[7] K. Singh, *Towards dynamic composition of question answering pipelines*, (2019) (cit. on p. 2).

[8] K. Singh, A. Both, A. Sethupat, and S. Shekarpour, "Frankenstein: A platform enabling reuse of question answering components," *European Semantic Web Conference*, Springer, 2018 624 (cit. on p. 2).

[9] K. Singh, A. Both, D. Diefenbach, S. Shekarpour, D. Cherix, and C. Lange, "Qanary–the fast track to creating a question answering system with linked data technology," *European Semantic Web Conference*, Springer, 2016 183 (cit. on p. 2).

[10] M. Dubey, *Towards Complex Question Answering over Knowledge Graphs*, (2021) (cit. on p. 2).

[11] S. Ruder, *An overview of multi-task learning in deep neural networks*, arXiv preprint arXiv:1706.05098 (2017) (cit. on pp. 2, 26, 27).

[12] D. Guo, D. Tang, N. Duan, M. Zhou, and J. Yin, "Dialog-to-action: Conversational question answering over a large-scale knowledge base," *Advances in Neural Information Processing Systems*, 2018 2942 (cit. on pp. 3, 62–65, 67, 72, 83).

[13] T. Shen, X. Geng, T. Qin, D. Guo, D. Tang, N. Duan, G. Long, and D. Jiang, “Multi-Task Learning for Conversational Question Answering over a Large-Scale Knowledge Base,” *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019 2442 (cit. on pp. 3, 63–65, 67, 71, 72, 76, 79, 83, 86, 90, 92).

[14] J. Plepi, E. Kacupaj, K. Singh, H. Thakkar, and J. Lehmann, “Context Transformer with Stacked Pointer Networks for Conversational Question Answering over Knowledge Graphs,” *European Semantic Web Conference*, Springer, 2021 356 (cit. on pp. 3, 4, 10, 24, 27, 44, 52, 62, 78, 90, 92).

[15] E. Kacupaj, J. Plepi, K. Singh, H. Thakkar, J. Lehmann, and M. Maleshkova, “Conversational Question Answering over Knowledge Graphs with Transformer and Graph Attention Networks,” *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, 2021 850 (cit. on pp. 3, 4, 10, 24, 25, 27, 44, 52, 62, 90–92).

[16] P. Christmann, R. Saha Roy, A. Abujabal, J. Singh, and G. Weikum, “Look before you hop: Conversational question answering over knowledge graphs using judicious context expansion,” *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019 729 (cit. on pp. 3, 30–32, 65, 72, 90–92, 99, 101–103).

[17] M. Kaiser, R. Saha Roy, and G. Weikum, “Reinforcement Learning from Reformulations in Conversational Question Answering over Knowledge Graphs,” *44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM, 2021 (cit. on pp. 3, 90–92, 96, 99, 102, 103).

[18] E. Kacupaj, K. Singh, M. Maleshkova, and J. Lehmann, “Contrastive Representation Learning for Conversational Question Answering over Knowledge Graphs,” *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022 925 (cit. on pp. 4, 11, 24, 27).

[19] W. Zheng, H. Cheng, L. Zou, J. X. Yu, and K. Zhao, “Natural language question/answering: Let users talk with the knowledge graph,” *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017 217 (cit. on pp. 4, 44).

[20] S. Ferré, *Sparklis: An expressive query builder for SPARQL endpoints with guidance in natural language*, Semantic Web **8** (2017) 405 (cit. on pp. 4, 44).

[21] D. Diefenbach, Y. Dridi, K. Singh, and P. Maret, “SPARQLtoUser: Did the question answering system understand me?” *ISWC 2017*, 2017 (cit. on pp. 4, 46).

[22] B. Ell, A. Harth, and E. Simperl, “SPARQL query verbalization for explaining semantic search engine queries,” *European Semantic Web Conference*, Springer, 2014 426 (cit. on pp. 4, 40, 44).

[23] A.-C. Ngonga Ngomo, L. Bühmann, C. Unger, J. Lehmann, and D. Gerber, "SPARQL2NL: verbalizing SPARQL queries," *Proceedings of the 22nd International Conference on World Wide Web*, 2013 329 (cit. on pp. 4, 40, 46).

[24] E. Kacupaj, H. Zafar, J. Lehmann, and M. Maleshkova, "Vquanda: Verbalization question answering dataset," *European Semantic Web Conference*, Springer, 2020 531 (cit. on pp. 5, 29–31, 34, 39, 44, 51, 54, 109).

[25] D. Biswas, M. Dubey, M. R. A. H. Rony, and J. Lehmann, *VANiLLa: Verbalized Answers in Natural Language at Large Scale*, arXiv preprint arXiv:2105.11407 (2021) (cit. on pp. 5, 29, 44, 51, 54, 100, 109).

[26] E. Kacupaj, B. Banerjee, K. Singh, and J. Lehmann, "ParaQA: A Question Answering Dataset with Paraphrase Responses for Single-Turn Conversation," *European Semantic Web Conference*, Springer, 2021 598 (cit. on pp. 9, 30, 44, 51, 54, 99, 100).

[27] E. Kacupaj, S. Premnadh, K. Singh, J. Lehmann, and M. Maleshkova, "VOGUE: Answer Verbalization Through Multi-Task Learning," *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2021 563 (cit. on pp. 9, 27).

[28] A. Singhal, *Introducing the knowledge graph: things, not strings*, Official google blog **5** (2012) 16 (cit. on p. 15).

[29] S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. Yu, *A Survey on Knowledge Graphs: Representation, Acquisition, and Applications.*, IEEE Transactions on Neural Networks and Learning Systems (2021) (cit. on p. 15).

[30] F. N. Stokman and P. H. de Vries, "Structuring knowledge in a graph," *Human-computer interaction*, Springer, 1988 186 (cit. on p. 15).

[31] A. Bordes, J. Weston, R. Collobert, and Y. Bengio, "Learning structured embeddings of knowledge bases," *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011 (cit. on p. 15).

[32] R. Cyganiak, D. Wood, M. Lanthaler, G. Klyne, J. J. Carroll, and B. McBride, *RDF 1.1 concepts and abstract syntax*, W3C recommendation **25** (2014) 1 (cit. on p. 15).

[33] H. Paulheim, *Knowledge graph refinement: A survey of approaches and evaluation methods*, Semantic web **8** (2017) 489 (cit. on p. 15).

[34] L. Ehrlinger and W. Wöß, *Towards a Definition of Knowledge Graphs.*, SEMANTiCS (Posters, Demos, SuCCESS) **48** (2016) 2 (cit. on p. 15).

[35] Q. Wang, Z. Mao, B. Wang, and L. Guo, *Knowledge graph embedding: A survey of approaches and applications*, IEEE Transactions on Knowledge and Data Engineering **29** (2017) 2724 (cit. on p. 15).

[36] S. Gottschalk, E. Kacupaj, S. Abdollahi, D. Alves, G. Amaral, E. Koutsiana, T. Kuculo, D. Major, C. Mello, G. S. Cheema, et al., "OEKG: The Open Event Knowledge Graph.," *CLEOPATRA@ WWW*, 2021 61 (cit. on pp. 15, 16).

[37] G. Tahmasebzadeh, E. Kacupaj, E. Müller-Budack, S. Hakimov, J. Lehmann, and R. Ewerth, "GeoWINE: Geolocation based Wiki, Image, News and Event Retrieval," *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021 2565 (cit. on pp. 15, 16).

[38] A. Guluzade, E. Kacupaj, and M. Maleshkova, "Demographic Aware Probabilistic Medical Knowledge Graph Embeddings of Electronic Medical Records," *International Conference on Artificial Intelligence in Medicine*, Springer, 2021 408 (cit. on pp. 15, 16).

[39] G. Zhu and C. A. Iglesias, "Sematch: Semantic Entity Search from Knowledge Graph.," *SumPre-HSWI@ ESWC*, 2015 (cit. on p. 16).

[40] I. O. Mulang', K. Singh, C. Prabhu, A. Nadgeri, J. Hoffart, and J. Lehmann, "Evaluating the impact of knowledge graph context on entity disambiguation models," *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020 2157 (cit. on pp. 16, 86).

[41] A. Sakor, I. O. Mulang, K. Singh, S. Shekarpour, M. E. Vidal, J. Lehmann, and S. Auer, "Old is gold: linguistic driven approach for entity and relation linking of short text," *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)*, 2019 2336 (cit. on p. 16).

[42] A. Sakor, K. Singh, A. Patel, and M.-E. Vidal, "Falcon 2.0: An entity and relation linking tool over Wikidata," *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020 3141 (cit. on p. 16).

[43] K. Singh, I. O. Mulang', I. Lytra, M. Y. Jaradeh, A. Sakor, M.-E. Vidal, C. Lange, and S. Auer, "Capturing knowledge in semantically-typed relational patterns to enhance relation linking," *Proceedings of the Knowledge Capture Conference*, 2017 1 (cit. on p. 16).

[44] X. Wang, X. He, Y. Cao, M. Liu, and T.-S. Chua, "Kgat: Knowledge graph attention network for recommendation," *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019 950 (cit. on p. 16).

[45] Z. Sun, J. Yang, J. Zhang, A. Bozzon, L.-K. Huang, and C. Xu, "Recurrent knowledge graph embedding for effective recommendation," *Proceedings of the 12th ACM Conference on Recommender Systems*, 2018 297 (cit. on p. 16).

[46] H. Wang, F. Zhang, M. Zhao, W. Li, X. Xie, and M. Guo, "Multi-task feature learning for knowledge graph enhanced recommendation," *The World Wide Web Conference*, 2019 2000 (cit. on p. 16).

[47] A. Ait-Mlouk and L. Jiang, *KBot: a Knowledge graph based chatBot for natural language understanding over linked data*, IEEE Access **8** (2020) 149220 (cit. on p. 16).

[48] S. Yoo and O. Jeong, *An Intelligent Chatbot Utilizing BERT Model and Knowledge Graph*, Journal of Society for e-Business Studies **24** (2020) (cit. on p. 16).

[49] Q. Bao, L. Ni, and J. Liu, “HHH: an online medical chatbot system based on knowledge graph and hierarchical bi-directional attention,” *Proceedings of the Australasian Computer Science Week Multiconference*, 2020 1 (cit. on p. 16).

[50] R. G. Athreya, A.-C. Ngonga Ngomo, and R. Usbeck, “Enhancing Community Interactions with Data-Driven Chatbots–The DBpedia Chatbot,” *Companion Proceedings of the The Web Conference 2018*, 2018 143 (cit. on p. 16).

[51] H. ter Horst and P. Cimiano, “Incorporating Semantic Dependencies Extracted from Knowledge Graphs into Joint Inference Template-Based Information Extraction,” *ECAI 2020*, IOS Press, 2020 2180 (cit. on p. 16).

[52] M. Zhou and V. Nastase, *Using patterns in knowledge graphs for targeted information extraction*, Proc. KBCOM (2018) (cit. on p. 16).

[53] T. Al-Moslmi, M. G. Ocaña, A. L. Opdahl, and C. Veres, *Named entity extraction for knowledge graphs: A literature overview*, IEEE Access **8** (2020) 32862 (cit. on p. 16).

[54] A. Nadgeri, A. Bastos, K. Singh, I. O. Mulang, J. Hoffart, S. Shekarpour, and V. Saraswat, *KGPool: Dynamic Knowledge Graph Context Selection for Relation Extraction*, arXiv preprint arXiv:2106.00459 (2021) (cit. on p. 16).

[55] M. P. K. Ravi, K. Singh, I. O. Mulang, S. Shekarpour, J. Hoffart, and J. Lehmann, “CHOLAN: A Modular Approach for Neural Entity Linking on Wikipedia and Wikidata,” *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, 2021 504 (cit. on p. 16).

[56] A. Hogan, E. Blomqvist, M. Cochez, C. d’Amato, G. D. Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier, et al., *Knowledge graphs*, ACM Computing Surveys (CSUR) **54** (2021) 1 (cit. on p. 16).

[57] A. Bordes, N. Usunier, S. Chopra, and J. Weston, *Large-scale simple question answering with memory networks*, arXiv preprint arXiv:1506.02075 (2015) (cit. on pp. 16, 31).

[58] L. Dong, F. Wei, M. Zhou, and K. Xu, “Question answering over freebase with multi-column convolutional neural networks,” *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)*, 2015 260 (cit. on pp. 16, 64).

[59] S. Hu, L. Zou, J. X. Yu, H. Wang, and D. Zhao, *Answering natural language questions by subgraph matching over knowledge graphs*, IEEE Transactions on Knowledge and Data Engineering **30** (2017) 824 (cit. on p. 16).

[60] Y. Lan, S. Wang, and J. Jiang, *Knowledge base question answering with topic units*, (2019) (cit. on p. 16).

[61] Y. Lan, S. Wang, and J. Jiang, *Knowledge base question answering with a matching-aggregation model and question-specific contextual relations*, IEEE/ACM Transactions on Audio, Speech, and Language Processing **27** (2019) 1629 (cit. on p. 16).

[62] S. Hu, L. Zou, and X. Zhang, "A state-transition framework to answer complex questions over knowledge base," *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018 2098 (cit. on p. 16).

[63] K. Luo, F. Lin, X. Luo, and K. Zhu, "Knowledge base question answering via encoding of complex query graphs," *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018 2185 (cit. on pp. 16, 64).

[64] Y. Lan, G. He, J. Jiang, J. Jiang, W. X. Zhao, and J.-R. Wen, *Complex Knowledge Base Question Answering: A Survey*, arXiv preprint arXiv:2108.06688 (2021) (cit. on p. 17).

[65] J. Berant and P. Liang, "Semantic parsing via paraphrasing," *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2014 1415 (cit. on p. 16).

[66] S. Reddy, M. Lapata, and M. Steedman, *Large-scale semantic parsing without question-answer pairs*, Transactions of the Association for Computational Linguistics **2** (2014) 377 (cit. on p. 16).

[67] K. Singh, M. Saleem, A. Nadgeri, F. Conrads, J. Z. Pan, A.-C. N. Ngomo, and J. Lehmann, "Qaldgen: Towards microbenchmarking of question answering systems over knowledge graphs," *International Semantic Web Conference*, Springer, 2019 277 (cit. on p. 16).

[68] S. Jain, "Question answering over knowledge base using factual memory networks," *Proceedings of the NAACL student research workshop*, 2016 109 (cit. on p. 17).

[69] Z.-Y. Chen, C.-H. Chang, Y.-P. Chen, J. Nayak, and L.-W. Ku, "UHop: An Unrestricted-Hop Relation Extraction Framework for Knowledge-Based Question Answering," *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)*, 2019 345 (cit. on p. 17).

[70] D. Bahdanau, K. Cho, and Y. Bengio, *Neural machine translation by jointly learning to align and translate*, arXiv preprint arXiv:1409.0473 (2014) (cit. on pp. 18–20, 25).

[71] T. Luong, H. Pham, and C. D. Manning, "Effective Approaches to Attention-based Neural Machine Translation," *EMNLP*, 2015 (cit. on pp. 18, 20, 38, 40, 54, 55, 102, 106).

[72] S. Srivastava, M. Patidar, S. Chowdhury, P. Agarwal, I. Bhattacharya, and G. Shroff, "Complex Question Answering on knowledge graphs using machine translation and multi-task learning," *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, 2021 3428 (cit. on p. 18).

[73] J. Bao, N. Duan, M. Zhou, and T. Zhao, "Knowledge-based question answering as machine translation," *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2014 967 (cit. on p. 18).

[74] Y. Liu and M. Lapata, "Text Summarization with Pretrained Encoders," *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019 3730 (cit. on p. 18).

[75] D. Aksenov, J. M. Schneider, P. Bourgonje, R. Schwarzenberg, L. Hennig, and G. Rehm, "Abstractive Text Summarization based on Language Model Conditioning and Locality Modeling," *Proceedings of the 12th Language Resources and Evaluation Conference*, 2020 6680 (cit. on p. 18).

[76] C. Huber, J. Hussain, T.-N. Nguyen, K. Song, S. Stüker, and A. Waibel, "Supervised Adaptation of Sequence-to-Sequence Speech Recognition Systems using Batch-Weighting," *Proceedings of the 2nd Workshop on Life-long Learning for Spoken Language Systems*, 2020 9 (cit. on p. 18).

[77] S. Kim, S. Dalmia, and F. Metze, "Gated Embeddings in End-to-End Speech Recognition for Conversational-Context Fusion," *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019 1131 (cit. on p. 18).

[78] I. Sutskever, O. Vinyals, and Q. V. Le, *Sequence to Sequence Learning with Neural Networks*, Advances in Neural Information Processing Systems **27** (2014) 3104 (cit. on pp. 18, 72).

[79] S. Hochreiter and J. Schmidhuber, *Long short-term memory*, Neural computation **9** (1997) 1735 (cit. on pp. 19, 20, 79).

[80] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014 1724 (cit. on pp. 19, 20).

[81] K. Fukushima and S. Miyake, "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition," *Competition and cooperation in neural nets*, Springer, 1982 267 (cit. on p. 20).

[82] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, 2017 5998 (cit. on pp. 21, 22, 25, 36, 38, 40, 47, 49, 50, 52, 54, 55, 69, 72, 76, 78, 79, 81, 83, 94, 99, 102, 106).

[83] X. Liu, K. Duh, L. Liu, and J. Gao, *Very deep transformers for neural machine translation*, arXiv preprint arXiv:2008.07772 (2020) (cit. on p. 23).

[84] J. Xu, J. Wang, M. Long, et al., *Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting*, Advances in Neural Information Processing Systems **34** (2021) (cit. on p. 23).

[85] Y. Liu and M. Lapata, “Hierarchical Transformers for Multi-Document Summarization,” *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019 5070 (cit. on p. 23).

[86] D. Lukovnikov, A. Fischer, and J. Lehmann, “Pretrained transformers for simple question answering over knowledge graphs,” *International Semantic Web Conference*, Springer, 2019 470 (cit. on p. 23).

[87] M. Arkhipov, M. Trofimova, Y. Kuratov, and A. Sorokin, “Tuning multilingual transformers for language-specific named entity recognition,” *Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing*, 2019 89 (cit. on p. 23).

[88] M. Ding, Z. Yang, W. Hong, W. Zheng, C. Zhou, D. Yin, J. Lin, X. Zou, Z. Shao, H. Yang, et al., *CogView: Mastering Text-to-Image Generation via Transformers*, arXiv preprint arXiv:2105.13290 (2021) (cit. on p. 23).

[89] J.-S. Bae, T.-J. Bak, Y.-S. Joo, and H.-Y. Cho, *Hierarchical Context-Aware Transformers for Non-Autoregressive Text to Speech*, arXiv preprint arXiv:2106.15144 (2021) (cit. on p. 23).

[90] Y. Wang, Z. Xu, X. Wang, C. Shen, B. Cheng, H. Shen, and H. Xia, “End-to-end video instance segmentation with transformers,” *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021 8741 (cit. on p. 23).

[91] H. Iuchi, T. Matsutani, K. Yamada, N. Iwano, S. Sumi, S. Hosoda, S. Zhao, T. Fukunaga, and M. Hamada, *Representation learning applications in biological sequence analysis*, Computational and Structural Biotechnology Journal **19** (2021) 3198 (cit. on p. 23).

[92] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” *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)*, 2019 4171 (cit. on pp. 23, 54, 55, 72, 96, 101).

[93] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, “BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension,” *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020 7871 (cit. on pp. 23, 24, 94, 96).

[94] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, *Improving language understanding by generative pre-training*, (2018) (cit. on pp. 23, 24).

[95] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, “Aligning books and movies: Towards story-like visual explanations by watching movies and reading books,” *Proceedings of the IEEE international conference on computer vision*, 2015 19 (cit. on p. 23).

[96] P. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, and K. kavukcuoglu, "Interaction networks for learning about objects, relations and physics," *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016 4509 (cit. on p. 24).

[97] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, and P. Battaglia, "Graph networks as learnable physics engines for inference and control," *International Conference on Machine Learning*, PMLR, 2018 4470 (cit. on p. 24).

[98] A. Fout, J. Byrd, B. Shariat, and A. Ben-Hur, "Protein interface prediction using graph convolutional networks," *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017 6533 (cit. on p. 24).

[99] Y. Wu, D. Lian, Y. Xu, L. Wu, and E. Chen, "Graph convolutional networks with markov random field reasoning for social spammer detection," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 01, 2020 1054 (cit. on p. 24).

[100] Z. Yang, W. Cohen, and R. Salakhudinov, "Revisiting semi-supervised learning with graph embeddings," *International conference on machine learning*, PMLR, 2016 40 (cit. on p. 24).

[101] T. Hamaguchi, H. Oiwa, M. Shimbo, and Y. Matsumoto, "Knowledge transfer for out-of-knowledge-base entities: a graph neural network approach," *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 2017 1802 (cit. on p. 24).

[102] H. Dai, E. B. Khalil, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017 6351 (cit. on p. 24).

[103] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, *Graph neural networks: A review of methods and applications*, AI Open **1** (2020) 57 (cit. on p. 25).

[104] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *International Conference on Learning Representations*, 2017 (cit. on p. 25).

[105] R. Li, S. Wang, F. Zhu, and J. Huang, "Adaptive Graph Convolutional Neural Networks," *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018 (cit. on p. 25).

[106] C. Zhuang and Q. Ma, "Dual graph convolutional networks for graph-based semi-supervised classification," *Proceedings of the 2018 World Wide Web Conference*, 2018 499 (cit. on p. 25).

[107] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," *International Conference on Learning Representations*, 2018 (cit. on pp. 25, 76, 80, 81).

[108] J. Zhang, X. Shi, J. Xie, H. Ma, I. King, and D. Y. Yeung, "GaAN: Gated Attention Networks for Learning on Large and Spatiotemporal Graphs," *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, 2018 (cit. on p. 25).

[109] R. Caruana, *Multitask learning*, Machine learning **28** (1997) 41 (cit. on pp. 26, 27).

[110] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," *Proceedings of the 25th international conference on Machine learning*, 2008 160 (cit. on p. 26).

[111] L. Deng, G. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: An overview," *2013 IEEE international conference on acoustics, speech and signal processing*, IEEE, 2013 8599 (cit. on p. 26).

[112] R. Girshick, "Fast r-cnn," *Proceedings of the IEEE international conference on computer vision*, 2015 1440 (cit. on p. 26).

[113] B. Ramsundar, S. Kearnes, P. Riley, D. Webster, D. Konerding, and V. Pande, *Massively Multitask Networks for Drug Discovery*, (2015) (cit. on p. 26).

[114] R. Caruana, "Multitask Learning: A Knowledge-Based Source of Inductive Bias," *Proceedings of the Tenth International Conference on Machine Learning*, Morgan Kaufmann, 1993 41 (cit. on p. 27).

[115] B. E. A. Boussaha, N. Hernandez, C. Jacquin, and E. Morin, *Deep retrieval-based dialogue systems: a short review*, arXiv preprint arXiv:1907.12878 (2019) (cit. on pp. 30, 31).

[116] M. Huang, X. Zhu, and J. Gao, *Challenges in building intelligent open-domain dialog systems*, ACM Transactions on Information Systems (TOIS) **38** (2020) 1 (cit. on p. 30).

[117] A. Saha, V. Pahuja, M. M. Khapra, K. Sankaranarayanan, and S. Chandar, "Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph," *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018 (cit. on pp. 30–32, 62–65, 67, 68, 71, 72, 76, 83, 84, 92, 99, 101).

[118] M. Dubey, D. Banerjee, A. Abdelkawi, and J. Lehmann, "Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia," *International semantic web conference*, Springer, 2019 69 (cit. on pp. 30–32).

[119] A. Abujabal, R. S. Roy, M. Yahya, and G. Weikum, "ComQA: A Community-sourced Dataset for Complex Factoid Question Answering with Paraphrase Clusters," *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)*, 2019 307 (cit. on pp. 30, 31).

[120] P. Trivedi, G. Maheshwari, M. Dubey, and J. Lehmann, "Lc-quad: A corpus for complex question answering over knowledge graphs," *International Semantic Web Conference*, Springer, 2017 210 (cit. on pp. 30–32, 34).

[121] Q. Cai and A. Yates,
"Large-scale semantic parsing via schema matching and lexicon extension,"
Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2013 423 (cit. on p. 31).

[122] J. Berant, A. Chou, R. Frostig, and P. Liang,
"Semantic parsing on freebase from question-answer pairs,"
Proceedings of the 2013 conference on empirical methods in natural language processing, 2013 1533 (cit. on pp. 31, 32).

[123] J. Bao, N. Duan, Z. Yan, M. Zhou, and T. Zhao,
"Constraint-based question answering with knowledge graph," *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2016 2503 (cit. on p. 31).

[124] Y. Su, H. Sun, B. Sadler, M. Srivatsa, I. Gür, Z. Yan, and X. Yan,
"On generating characteristic-rich question sets for qa evaluation,"
Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 2016 562 (cit. on p. 31).

[125] A. Talmor and J. Berant,
"The Web as a Knowledge-Base for Answering Complex Questions,"
Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), 2018 641 (cit. on p. 31).

[126] W.-t. Yih, M. Richardson, C. Meek, M.-W. Chang, and J. Suh,
"The value of semantic parse labeling for knowledge base question answering,"
Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), 2016 201 (cit. on pp. 31, 64).

[127] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor,
"Freebase: a collaboratively created graph database for structuring human knowledge,"
Proceedings of the 2008 ACM SIGMOD international conference on Management of data, 2008 1247 (cit. on p. 31).

[128] R. Lowe, N. Pow, I. V. Serban, and J. Pineau, "The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems," *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, 2015 285 (cit. on p. 31).

[129] S. Zamanirad, B. Benatallah, C. Rodriguez, M. Yaghoubzadehfard, S. Bouguelia, and H. Brabra, "State machine based human-bot conversation model and services," *International Conference on Advanced Information Systems Engineering*, Springer, 2020 199 (cit. on p. 31).

[130] I. V. Serban, A. Garcia-Duran, C. Gulcehre, S. Ahn, S. Chandar, A. Courville, and Y. Bengio, "Generating Factoid Questions With Recurrent Neural Networks: The 30M Factoid Question-Answer Corpus," *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016 588 (cit. on p. 32).

[131] K. McKeown, *Paraphrasing questions using given and new information*, American Journal of Computational Linguistics **9** (1983) 1 (cit. on p. 32).

[132] C. Quirk, C. Brockett, and W. B. Dolan, "Monolingual machine translation for paraphrase generation," *Proceedings of the 2004 conference on empirical methods in natural language processing*, 2004 142 (cit. on p. 32).

[133] S. Wubben, A. Van Den Bosch, and E. Krahmer, "Paraphrase generation as monolingual translation: Data and evaluation," *Proceedings of the 6th International Natural Language Generation Conference*, 2010 (cit. on p. 32).

[134] A. Prakash, S. A. Hasan, K. Lee, V. Datla, A. Qadir, J. Liu, and O. Farri, "Neural Paraphrase Generation with Stacked Residual LSTM Networks," *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2016 2923 (cit. on p. 32).

[135] S. A. Hasan, B. Liu, J. Liu, A. Qadir, K. Lee, V. Datla, A. Prakash, and O. Farri, "Neural clinical paraphrase generation with attention," *Proceedings of the Clinical Natural Language Processing Workshop (ClinicalNLP)*, 2016 42 (cit. on p. 32).

[136] E. Egonmwan and Y. Chali, "Transformer and seq2seq model for paraphrase generation," *Proceedings of the 3rd Workshop on Neural Generation and Translation*, 2019 249 (cit. on p. 32).

[137] M. Honnibal and I. Montani, *Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing*, Unpublished software application. https://spacy. io (2017) (cit. on p. 34).

[138] S. Edunov, M. Ott, M. Auli, and D. Grangier, "Understanding Back-Translation at Scale," *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018 489 (cit. on pp. 36, 99).

[139] C. Federmann, O. Elachqar, and C. Quirk, "Multilingual whispers: Generating paraphrases with translation," *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, 2019 17 (cit. on pp. 36, 99).

[140] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," *International Conference on Machine Learning*, PMLR, 2017 1243 (cit. on pp. 38, 40, 54, 55, 102, 106).

[141] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002 311 (cit. on pp. 38, 54).

[142] S. Banerjee and A. Lavie, "METEOR: An automatic metric for MT evaluation with improved correlation with human judgments," *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, 2005 65 (cit. on pp. 39, 54).

[143] P. Gupta, S. Mehri, T. Zhao, A. Pavel, M. Eskenazi, and J. P. Bigham, "Investigating Evaluation of Open-Domain Dialogue Systems With Human Generated Multiple References," *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, 2019 379 (cit. on p. 39).

[144] A.-C. Ngonga Ngomo, L. Bühmann, C. Unger, J. Lehmann, and D. Gerber, "Sorry, i don't speak SPARQL: translating SPARQL queries into natural language," *Proceedings of the 22nd international conference on World Wide Web*, 2013 977 (cit. on p. 40).

[145] B. Fu, Y. Qiu, C. Tang, Y. Li, H. Yu, and J. Sun, *A survey on complex question answering over knowledge base: Recent advances and challenges*, arXiv preprint arXiv:2007.13069 (2020) (cit. on pp. 44, 90).

[146] K. Singh, I. Lytra, A. S. Radhakrishna, S. Shekarpour, M.-E. Vidal, and J. Lehmann, *No one is perfect: Analysing the performance of question answering components over the dbpedia knowledge graph*, Journal of Web Semantics **65** (2020) 100594 (cit. on p. 44).

[147] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018 7482 (cit. on pp. 44, 50, 82).

[148] T.-H. Wen, D. Vandyke, N. Mrkšić, M. Gasic, L. M. R. Barahona, P.-H. Su, S. Ultes, and S. Young, "A Network-based End-to-End Trainable Task-oriented Dialogue System," *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, 2017 438 (cit. on p. 45).

[149] A. Bordes, Y.-L. Boureau, and J. Weston, *Learning End-to-End Goal-Oriented Dialog*, (2016) (cit. on p. 45).

[150] A. Madotto, C.-S. Wu, and P. Fung, "Mem2Seq: Effectively Incorporating Knowledge Bases into End-to-End Task-Oriented Dialog Systems," *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018 1468 (cit. on p. 45).

[151] M. Eric, L. Krishnan, F. Charette, and C. D. Manning, "Key-Value Retrieval Networks for Task-Oriented Dialogue," *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, 2017 37 (cit. on p. 45).

[152] W. Lei, X. Jin, M.-Y. Kan, Z. Ren, X. He, and D. Yin, "Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures," *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018 1437 (cit. on p. 46).

[153] J. Gu, Z. Lu, H. Li, and V. O. Li, "Incorporating Copying Mechanism in Sequence-to-Sequence Learning," *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016 1631 (cit. on p. 46).

[154] F. Kassawat, D. Chaudhuri, and J. Lehmann,
"Incorporating joint embeddings into goal-oriented dialogues with multi-task learning,"
European Semantic Web Conference, Springer, 2019 225 (cit. on pp. 46, 65).

[155] C. Gardent, A. Shimorina, S. Narayan, and L. Perez-Beltrachini,
"Creating Training Corpora for NLG Micro-Planners," *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017 179 (cit. on p. 46).

[156] H. Gao, L. Wu, P. Hu, and F. Xu,
"RDF-to-Text Generation with Graph-augmented Structural Neural Encoders.," *IJCAI*, 2020 3030 (cit. on p. 46).

[157] C. Zhao, M. Walker, and S. Chaturvedi,
"Bridging the structural gap between encoding and decoding for data-to-text generation,"
Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020 2481 (cit. on p. 46).

[158] L. Song, A. Wang, J. Su, Y. Zhang, K. Xu, Y. Ge, and D. Yu,
"Structural Information Preserving for Graph-to-Text Generation,"
Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020 7987 (cit. on p. 46).

[159] J. Liu, S. Chen, B. Wang, J. Zhang, N. Li, and T. Xu, "Attention as Relation: Learning Supervised Multi-head Self-Attention for Relation Extraction.," *IJCAI*, 2020 3787 (cit. on p. 46).

[160] X. Shen, E. Chang, H. Su, C. Niu, and D. Klakow,
"Neural Data-to-Text Generation via Jointly Learning the Segmentation and Correspondence,"
Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020 7155 (cit. on p. 46).

[161] I. Yamada, A. Asai, H. Shindo, H. Takeda, and Y. Matsumoto,
"LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention,"
Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020 6442 (cit. on p. 47).

[162] J. Pennington, R. Socher, and C. D. Manning,
"Glove: Global vectors for word representation," *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014 1532 (cit. on pp. 47, 69, 78).

[163] X. Wei, T. Zhang, Y. Li, Y. Zhang, and F. Wu,
"Multi-modality cross attention network for image and sentence matching,"
Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020 10941 (cit. on p. 49).

[164] S. Mohla, S. Pande, B. Banerjee, and S. Chaudhuri, "Fusatnet: Dual attention based spectrospatial multimodal fusion network for hyperspectral and lidar classification,"
Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020 92 (cit. on p. 49).

[165] J. Armitage, E. Kacupaj, G. Tahmasebzadeh, M. Maleshkova, R. Ewerth, and J. Lehmann, "Mlm: A benchmark dataset for multitask learning with multiple languages and modalities," *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020 2967 (cit. on pp. 50, 82).

[166] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *ICLR (Poster)*, 2015 (cit. on pp. 52, 83).

[167] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, *Dropout: a simple way to prevent neural networks from overfitting*, The journal of machine learning research **15** (2014) 1929 (cit. on pp. 52, 83).

[168] D. Fensel, U. Şimşek, K. Angele, E. Huaman, E. Kärle, O. Panasiuk, I. Toma, J. Umbrich, and A. Wahler, "Why we need knowledge graphs: Applications," *Knowledge Graphs*, Springer, 2020 95 (cit. on p. 62).

[169] K. Bollacker, R. Cook, and P. Tufts, "Freebase: A shared database of structured general human knowledge," *AAAI*, vol. 7, 2007 1962 (cit. on p. 62).

[170] R. Jia and P. Liang, "Data Recombination for Neural Semantic Parsing," *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016 12 (cit. on p. 62).

[171] C. Liang, J. Berant, Q. Le, K. Forbus, and N. Lao, "Neural Symbolic Machines: Learning Semantic Parsers on Freebase with Weak Supervision," *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017 23 (cit. on pp. 62–64).

[172] L. Dong and M. Lapata, "Coarse-to-Fine Decoding for Neural Semantic Parsing," *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018 731 (cit. on p. 62).

[173] C. Xiao, M. Dymetman, and C. Gardent, "Sequence-based structured prediction for semantic parsing," *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016 1341 (cit. on p. 63).

[174] L. Dong and M. Lapata, "Language to Logical Form with Neural Attention," *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016 33 (cit. on p. 63).

[175] Y. Zhang, H. Dai, Z. Kozareva, A. J. Smola, and L. Song, "Variational reasoning for question answering with knowledge graph," *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018 (cit. on pp. 63, 64).

[176] K. Xu, S. Reddy, Y. Feng, S. Huang, and D. Zhao, "Question Answering on Freebase via Relation Extraction and Textual Evidence," *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016 2326 (cit. on p. 63).

[177] J. Gao, M. Galley, and L. Li, “Neural approaches to conversational ai,” *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2018 1371 (cit. on p. 64).

[178] D. Lukovnikov, A. Fischer, J. Lehmann, and S. Auer, “Neural network-based question answering over knowledge graphs on word and character level,” *Proceedings of the 26th international conference on World Wide Web*, 2017 1211 (cit. on p. 64).

[179] I. Serban, A. Sordoni, Y. Bengio, A. Courville, and J. Pineau, “Building end-to-end dialogue systems using generative hierarchical neural network models,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, 1, 2016 (cit. on pp. 64, 72, 92).

[180] A. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston, “Key-Value Memory Networks for Directly Reading Documents,” *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016 1400 (cit. on pp. 64, 72, 92).

[181] P. Marion, P. K. Nowak, and F. Piccinno, *Structured Context and High-Coverage Grammar for Conversational Question Answering over Knowledge Graphs*, arXiv preprint arXiv:2109.00269 (2021) (cit. on pp. 64, 102).

[182] R. Thirukovalluru, M. Sridhar, D. Thai, S. Chanumolu, N. Monath, S. Ananthakrishnan, and A. McCallum, “Knowledge Informed Semantic Parsing for Conversational Question Answering,” *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, 2021 231 (cit. on p. 65).

[183] S. Vakulenko, S. Longpre, Z. Tu, and R. Anantha, “Question rewriting for conversational question answering,” *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 2021 355 (cit. on pp. 65, 72).

[184] O. Vinyals, M. Fortunato, and N. Jaitly, “Pointer networks,” *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*, 2015 2692 (cit. on pp. 65, 70, 71, 95).

[185] W. Lu, H. T. Ng, W. S. Lee, and L. Zettlemoyer, “A generative model for parsing natural language to meaning representations,” *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, 2008 783 (cit. on p. 67).

[186] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus, “End-to-end memory networks,” *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*, 2015 2440 (cit. on p. 72).

[187] Z. Li, Y. Zhang, Y. Wei, Y. Wu, and Q. Yang, “End-to-End Adversarial Memory Network for Cross-domain Sentiment Classification.,” *IJCAI*, 2017 2237 (cit. on p. 72).

[188] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al., *Google's neural machine translation system: Bridging the gap between human and machine translation*, arXiv preprint arXiv:1609.08144 (2016) (cit. on p. 78).

[189] I. O. Mulang, K. Singh, A. Vyas, S. Shekarpour, M.-E. Vidal, J. Lehmann, and S. Auer, "Encoding knowledge graph entity aliases in attentive neural network for wikidata entity linking," *International Conference on Web Information Systems Engineering*, Springer, 2020 328 (cit. on p. 86).

[190] M. Zaib, W. E. Zhang, Q. Z. Sheng, A. Mahmood, and Y. Zhang, *Conversational Question Answering: A Survey*, arXiv preprint arXiv:2106.00874 (2021) (cit. on pp. 90, 92).

[191] K. Singh, A. S. Radhakrishna, A. Both, S. Shekarpour, I. Lytra, R. Usbeck, A. Vyas, A. Khikmatullaev, D. Punjani, C. Lange, et al., "Why reinvent the wheel: Let's build question answering systems together," *Proceedings of the 2018 World Wide Web Conference*, 2018 1247 (cit. on p. 90).

[192] A. Einolghozati, A. Gupta, K. Diedrick, and S. Gupta, "Sound Natural: Content Rephrasing in Dialog Systems," *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020 5101 (cit. on pp. 90, 99, 100).

[193] D. Peskov, N. Clarke, J. Krone, B. Fodor, Y. Zhang, A. Youssef, and M. Diab, "Multi-domain goal-oriented dialogues (multidogo): Strategies toward curating and annotating large scale dialogue data," *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019 4526 (cit. on p. 90).

[194] A. Acharya, S. Adhikari, S. Agarwal, V. Auvray, N. Belgamwar, A. Biswas, S. Chandra, T. Chung, M. Fazel-Zarandi, R. Gabriel, et al., "Alexa Conversations: An Extensible Data-driven Approach for Building Task-oriented Dialogue Systems," *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstrations*, 2021 125 (cit. on p. 90).

[195] T. F. Doyle, *The role of context in meaning and understanding*, PhD thesis: Universitaet Potsdam, 2007 (cit. on p. 91).

[196] J. J. Lin, D. Quan, V. Sinha, K. Bakshi, D. Huynh, B. Katz, and D. R. Karger, "What Makes a Good Answer? The Role of Context in Question Answering.," *INTERACT*, vol. 3, Citeseer, 2003 25 (cit. on p. 91).

[197] H. Bast and E. Haussmann, "More accurate question answering on freebase," *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 2015 1431 (cit. on p. 92).

[198] W.-t. Yih, M.-W. Chang, X. He, and J. Gao, "Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base," *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)*, 2015 1321 (cit. on p. 92).

[199] M. Yu, W. Yin, K. S. Hasan, C. dos Santos, B. Xiang, and B. Zhou, "Improved Neural Relation Detection for Knowledge Base Question Answering," *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017 571 (cit. on p. 92).

[200] A. Parikh, O. Täckström, D. Das, and J. Uszkoreit, "A Decomposable Attention Model for Natural Language Inference," *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016 2249 (cit. on p. 92).

[201] G. Maheshwari, P. Trivedi, D. Lukovnikov, N. Chakraborty, A. Fischer, and J. Lehmann, "Learning to rank query graphs for complex question answering over knowledge graphs," *International semantic web conference*, Springer, 2019 487 (cit. on p. 92).

[202] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al., *Language models are unsupervised multitask learners*, OpenAI blog **1** (2019) 9 (cit. on p. 94).

[203] A. Radford, J. Wu, D. Amodei, D. Amodei, J. Clark, M. Brundage, and I. Sutskever, *Better language models and their implications*, OpenAI Blog https://openai. com/blog/better-language-models **1** (2019) 2 (cit. on p. 96).

[204] E. Kacupaj, K. Singh, M. Maleshkova, and J. Lehmann, *An Answer Verbalization Dataset for Conversational Question Answerings over Knowledge Graphs*, arXiv preprint arXiv:2208.06734 (2022) (cit. on p. 99).

[205] O. Bojar, R. Chatterjee, C. Federmann, M. Fishel, Y. Graham, B. Haddow, M. Huck, A. J. Yepes, P. Koehn, C. Monz, et al., "Proceedings of the Third Conference on Machine Translation," *Proceedings of the Third Conference on Machine Translation: Research Papers*, 2018 (cit. on p. 99).

[206] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," *International Conference on Learning Representations*, 2018 (cit. on p. 101).

[207] P. Kouki, J. Schaffer, J. Pujara, J. O'Donovan, and L. Getoor, "User preferences for hybrid explanations," *Proceedings of the Eleventh ACM Conference on Recommender Systems*, 2017 84 (cit. on p. 113).

[208] S. Shekarpour, A. Nadgeri, and K. Singh, *QA2Explanation: Generating and Evaluating Explanations for Question Answering Systems over Knowledge Graph*, arXiv preprint arXiv:2010.08323 (2020) (cit. on p. 113).

APPENDIX A

List of Publications

- ***Conference Papers (peer reviewed)***

  1. **Endri Kacupaj**, Joan Plepi, Kuldeep Singh, Harsh Thakkar, Jens Lehmann, and Maria Maleshkova. "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, pp. 850-862. 2021. DOI: 10.18653/v1/2021.eacl-main.72 (part of the dissertation)
  2. **Endri Kacupaj**, Shyamnath Premnadh, Kuldeep Singh, Jens Lehmann, and Maria Maleshkova. "VOGUE: Answer Verbalization Through Multi-Task Learning." In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 563-579. Springer, Cham, 2021. DOI: 10.1007/978-3-030-86523-8_34 (part of the dissertation)
  3. Joan Plepi, **Endri Kacupaj**, Kuldeep Singh, Harsh Thakkar, and Jens Lehmann. "Context Transformer with Stacked Pointer Networks for Conversational Question Answering over Knowledge Graphs." In European Semantic Web Conference, pp. 356-371. Springer, Cham, 2021. DOI: 10.1007/978-3-030-77385-4_21 (part of the dissertation)
  4. **Endri Kacupaj**, Barshana Banerjee, Kuldeep Singh, and Jens Lehmann. "ParaQA: A Question Answering Dataset with Paraphrase Responses for Single-Turn Conversation." In European Semantic Web Conference, pp. 598-613. Springer, Cham, 2021. DOI: 10.1007/978-3-030-77385-4_36 (part of the dissertation)
  5. **Endri Kacupaj**, Hamid Zafar, Jens Lehmann, and Maria Maleshkova. "VQuAnDa: Verbalization question answering dataset." In European Semantic Web Conference, pp. 531-547. Springer, Cham, 2020. DOI: 10.1007/978-3-030-49461-2_31
  6. **Endri Kacupaj**, Kuldeep Singh, Maria Maleshkova, and Jens Lehmann. "Contrastive Representation Learning for Conversational Question Answering over Knowledge Graphs." In Proceedings of the 31st ACM International Conference on Information & Knowledge Management, pp. 925-934. 2022. DOI: 10.1145/3511808.3557267 (part of the dissertation)
  7. Jason Armitage, **Endri Kacupaj**, Golsa Tahmasebzadeh, Maria Maleshkova, Ralph Ewerth, and Jens Lehmann. "MLM: A benchmark dataset for multitask learning with

multiple languages and modalities." In Proceedings of the 29th ACM International Conference on Information & Knowledge Management, pp. 2967-2974. 2020. DOI: 10.1145/3340531.3412783

- ***Demo Papers (peer reviewed)***

8. Golsa Tahmasebzadeh, **Endri Kacupaj**, Eric Müller-Budack, Sherzod Hakimov, Jens Lehmann, and Ralph Ewerth. 2021. "GeoWINE: Geolocation based Wiki, Image, News and Event Retrieval." In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21). Association for Computing Machinery, New York, NY, USA, 2565–2569. DOI: 10.1145/3404835.3462786

- ***Workshop Articles (peer reviewed)***

9. Simon Gottschalk, **Endri Kacupaj**, Sara Abdollahi, Diego Alves, Gabriel Amaral, Elisavet Koutsiana, Tin Kuculo et al. "OEKG: The Open Event Knowledge Graph." In CLEOPATRA@ WWW, pp. 61-75. 2021. DOI: Vol-2829/paper5

- ***Miscellaneous Papers (peer reviewed)***

Following publication originated during the thesis but is not part of the thesis itself.

10. Aynur Guluzade, **Endri Kacupaj**, and Maria Maleshkova. "Demographic Aware Probabilistic Medical Knowledge Graph Embeddings of Electronic Medical Records." In International Conference on Artificial Intelligence in Medicine, pp. 408-417. Springer, Cham, 2021. DOI: 10.1007/978-3-030-77211-6_48

List of Figures

List of Tables