THE UNIVERSITY *of* EDINBURGH

Explicit Context Representations for Conversational Language Understanding

Parag Jain

Doctor of Philosophy

Institute for Language, Cognition and Computation
School of Informatics
University of Edinburgh

2025

Abstract

Natural language interfaces enable users to interact with automated systems by using human languages, such as English. Users often pose complex or incomplete questions as they explore and refine their needs through conversational queries, such as, *Where did Christopher Bishop earn his PhD? – Edinburgh – Where is it located?*. Effectively responding to such queries requires the system to understand conversational language. This entails resolving references, handling topic-extensions and topic-shift, and managing long-term dependencies, all situated within the specific context of the ongoing exchange.

In this thesis, we are interested in building systems capable of conversational language understanding, which is fundamental to multiple tasks such as question answering, task completion (e.g., hotel booking, calendar management), and semantic parsing. Such a system necessitates the integration of both *discourse* and *external* contexts. The discourse context includes earlier interactions, such as previous queries and responses. The external context is task-dependent but is also critical; for instance, a ticket booking system needs to access the latest flight schedules, while a question-answering system might require access to knowledge from external sources like Wikipedia or Wikidata. As the length of interaction grows, maintenance and utilization of relevant context information become crucial to effectively addressing user queries.

This thesis argues that **maintaining explicit representations of context – external or discourse – is helpful for tasks requiring long-range interactions**. We show that dynamic representations are both computationally efficient in representing context and effective across various conversational tasks. First, we look at the problem of modeling discourse context for long-range interactions and propose to represent discourse information using a bounded external memory. Our memory is end-to-end learned and interpretable. On the task of conversational semantic parsing over relational databases we show that our method is able to maintain performance over long-range interactions and can handle several discourse related phenomena such as topic-shift and handling referring expressions. Second, we look at the problem of modeling external context in the form of a knowledge graph. We create SPICE, a semantic parsing dataset for conversational question answering over Wikidata knowledge graph (KG). Wikidata contains millions of entities and thousands of relations making it infeasible to fully encode the KG. To tackle this, we propose dynamic context graphs that represent information about user utterance and its context through a dynamically generated subgraph, wherein the number of nodes varies for each utterance. We further show that the graph-structure

itself is crucial, and merely linearizing the subgraph as a textual string is neither efficient nor performant. These findings establish that structured representations of context are beneficial. Finally, we validate our hypothesis on tasks other than semantic parsing with large pretrained language models. We focus on conversational question answering over heterogeneous sources. We employ a graph-structured representation to unify information from different sources, such as, text, knowledge graphs, tables, and infoboxes. Our approach maintains a conversational memory to track and update past evidence, thus influencing the graph's structure and representation, as the conversation evolves. We show that the graph enhances the language model's ability to reason over multiple modalities, while the memory module provides robustness against noise and retrieval errors.

Lay Summary

Conversational agents like Alexa offer a natural language interface for interacting with digital systems. Beyond basic functionalities like setting alarms, more complex tasks require multi-turn interactions that also need connecting with various data sources. For instance, a user might ask an automated travel agent about flights, then inquire about baggage policies, followed by questions on tourist attractions. To handle these interactions, the system must connect to various data contexts such as flight databases, airline policy documents, and Wikipedia for general knowledge, all while keeping track of the context of the conversation. This requires systems to recall previous exchanges and access information in real-time to respond accurately to the user's most recent query.

In this thesis, we focus on building systems that are capable of processing multi-turn conversations for tasks that require connecting with external data sources (e.g., relational database, text documents). To process multi-turn queries such as, *Which airlines operate flights from Edinburgh to London? – British Airways and Qatar Airways. – Which one offers early morning flights?*. A hypothetical system needs to resolve references (i.e, computing what does *Which one* refers to), handle change in topic (for example, moving from *flights* to *tourist attractions*), and manage such dependencies for over extended conversations. Hence it must consider both the conversation history (previous query response pairs) and external information (like a flight database). As conversations grows longer, keeping track of relevant information becomes key to effectively addressing user needs.

We consider different strategies for modeling various types of contextual information in conversational agents. We introduce a method for managing long conversations using a small memory that stores relevant parts of the conversational history. We explore how to integrate information from different external data sources, such as a database of flight records or general knowledge in the form of a graph. We propose a method to integrate information from multiple sources into recent large language models. Our methods access facts in real-time, making them suitable for real-world applications. The proposed solutions maintain performance over long interactions and can handle referring expressions and topic-shifts. Our contributions improve conversational systems by leveraging the underlying structure of conversational and external context.

Acknowledgements

Countless individuals have played a part in my PhD journey, and I am forever grateful for their support, guidance, and friendship.

I am profoundly grateful to my supervisor, Mirella Lapata, for her mentorship throughout my PhD. Her guidance on writing and presenting ideas has been invaluable, helping me continuously refine my work and communicate my research effectively. Mirella's kindness and patience, especially through the challenges and rejections, have been a source of constant encouragement. I'm thankful to Alex Lascarides, Ivan Titov, and Pasquale Minervini for their thoughtful feedback during the yearly reviews. Their suggestions have significantly improved the quality of my research. I would also like to extend a heartfelt thanks to Mark Steedman for his insightful discussions and encouragement during initial period of my PhD.

I am grateful to Ratish and Yumo for their many discussions and willingness to provide suggestions whenever I felt stuck. I am indebted to my CDT NLP cohort for their support and friendship and would also like to express my gratitude to Rohit and Nikita for their friendship and lending their ears for on-going rants. A special thanks to Sally for her efforts in ensuring that we made the most of our time in the programme.

I would not have skills to pursue a PhD without the initial trust that Karthik showed while hiring me without any experience in NLP. I would like to acknowledge the friendship and mentorship that Anirban, Abhijit and Amar offered. I have learned the most by interacting with them, their patience in guiding me through the process have been instrumental in my academic success.

I would like to thank my family for their love, support, and encouragement. I would not have courage to leave my job and start a PhD without the support of my brothers. I will be eternally thankful to them. My heartfelt gratitude goes to my wife, Nidhi, who shared this journey with me. She moved to a new country without a second thought, selflessly prioritizing my path over her own. Her steadfast support through the good, the bad, and the ugly has been truly immeasurable.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Parag Jain*)

Table of Contents

Chapter 1

Introduction

1.1 Motivation

As technology becomes more intuitive and personal, people are changing the way they interact with it. It is natural to ask follow-up questions when using voice assistants, making conversational language processing increasingly important for the success of natural language interfaces. Automated systems must access various information sources and take appropriate actions based on users' information needs. Context, which encompasses relevant background information and situational variables, plays a crucial role in shaping these interactions. Theoretically, context is of infinite dimension (McCarthy, 1993), and hence cannot be specified explicitly. However, it is natural for humans to distill situations, identify key aspects, and bound the context spontaneously. Therefore, it is clear that not all aspects of context need to be defined and processed in real-world scenarios to take effective action.

Context plays an important role in communication and computing. In communication, it helps interlocutors communicate efficiently without specifying every single detail (Grice, 1975). As a result utterances such as *Please close the door when you leave* can be understood despite the absence of additional details (e.g., which *door*?). In computing, context helps define the exact scope of the application at hand. For example, in human computer interfaces, context defines the identity, location and the objects involved (Dey, 2001). A query such as *Start navigation to the office*, to Google Maps, can be fulfilled without further enquiry by using the user's location and personal context to implicitly identify both the starting position and the destination, *office*.

Consider an automated travel agent that enables users to create travel plans. Example interaction in Figure 1.1 shows user queries about traveling to London, which

involve reasoning through various sources of information to meet their needs. The user queries vary widely and require interacting with diverse information sources to generate appropriate responses. For instance, to address the user query *I need a flight from Edinburgh to London*, the agent must connect to a flight booking database to find available options. This necessitates parsing the utterance into a logical form, such as SQL, that can be executed on the database (Androutsopoulos et al., 1995; Zelle and Mooney, 1996; Miller et al., 1996). While the question *What are the luggage policies for these flights?* can be answered by retrieving relevant information from airline documentation and reasoning to extract the desired details (Dalton et al., 2020). Given that such documents are typically semi-structured, it further demand reasoning across multiple textual modalities (e.g., tables). Here, the phrase *these flights* demonstrates coreference, linking back to the specific flights mentioned in the prior query, which needs the agent to maintain contextual awareness (Mitkov, 1999; Castagnola, 2002). The follow-on query *Check my past flight bookings to see if I have any loyalty points I can use?* is similar to the initial query and requires accessing a database via parsing the query to SQL to with updated users information. Final query *What are some popular attractions in London?* in addition mandates access to an external general knowledge source, such as Wikidata (Vrandečić and Krötzsch, 2014), which can be queried via SPARQL.

This example highlights that users often pose complex or incomplete questions as they explore and refine their needs through conversational queries. To effectively address these query, a machine learning system must account for both the preceding *discourse* context, which includes earlier interactions, and the broader *external* context relevant to the dialogue. The *external* context varies by task and is critical for effectively responding to different queries which in turn can be answered though access to different types of contextual information. In this thesis, we are interested in modeling context explicitly, such that it allows for effective long-range interactions. We propose context-centric models for different tasks that are fundamental to a natural language interface system. We empirically evaluate the significance of explicit context management across a range of conversational tasks, including conversational semantic parsing and question answering.

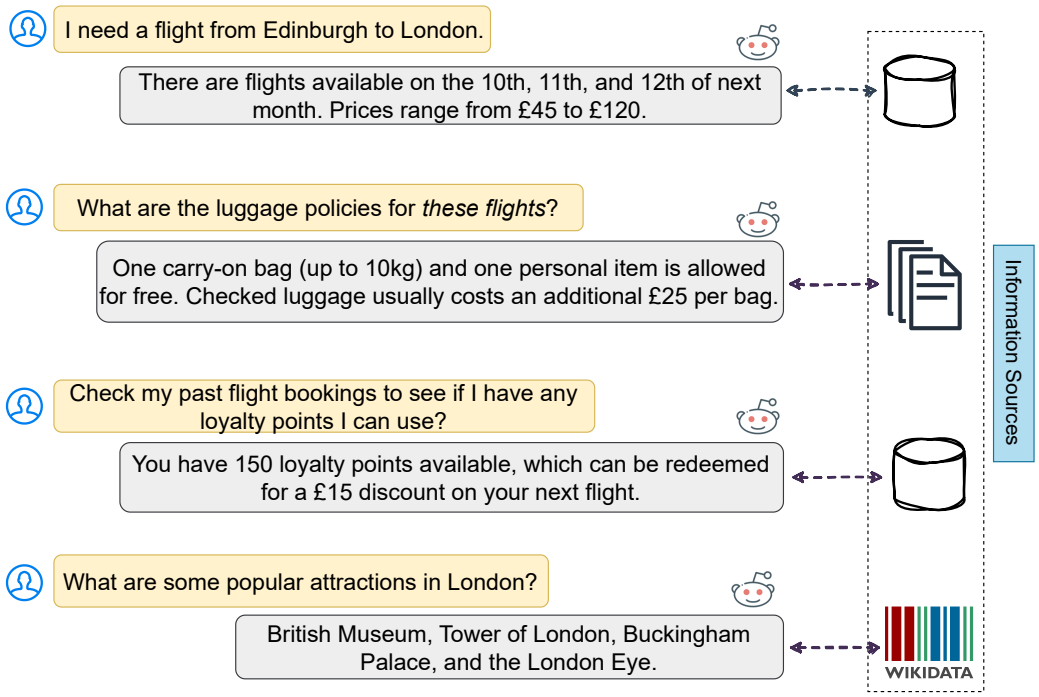

Figure 1.1: An interaction with an automated travel agent, highlighting various user queries that demand engagement with diverse information sources to generate appropriate responses. The first query, *I need a flight from...,* and the third query, *Check my past flight bookings...,* require access to a relational database containing flight information and users' personal records, which can be queried using languages such as SQL. Depending on the design, these databases may be two separate instances with different underlying schemas. The second query, *What are the luggage...,* necessitates access to airline documents, which are typically semi-structured and require reasoning across multiple textual modalities. The final query, *What are some...,* requires access to a general-purpose knowledge base such as Wikidata (Vrandečić and Krötzsch, 2014), that is accessible by parsing the utterance into a formal representation, such as SPARQL. Overall, this example interaction demonstrates that a real-world automated agent must be able to interact with various types of *external* context, and handle extended interactions.

1.2 Thesis Outline

Our interest in this thesis is in building systems capable of conversational language understanding. Our primary hypothesis is that **maintaining explicit representations of context – external or discourse – is helpful for tasks requiring long-range interactions.** We investigate this hypothesis by answering the following questions:

- Q1. How can we model discourse context effectively for long-range interactions? – (Chapter 3)
- Q2: How can we represent external context (e.g., a knowledge graph) when it is infeasible to encode it within the model due to its large scale? – (Chapter 4 and 5)
- Q3: To what extent do our findings apply to large language models pretrained on web-scale data? – (Chapter 6)

We explore these questions in rest of the thesis that is organized into the following chapters.

Background In Chapter 2, we provide an overview of the background relevant to our work in conversational question answering. We discuss different types of context considered and formulate specific applications explored in this thesis namely, semantic parsing and direct response generation. Additionally, we describe the base modeling architectures employed and the evaluation metrics used. Finally, we present statistics and examples of the various datasets utilized throughout the thesis.

Modeling Discourse Context In Chapter 3, we explore the question of how to incorporate discourse context that is effective for long-range interactions. We propose to represent discourse information using a bounded external memory, i.e., our memory uses constant space irrespective of the length of the conversation. This helps process long interactions irrespective of conversation-length. We also interpret the learned controller and memory contents, and show one-to-one correspondence between the controller's decision and the output quality. Our results in three conversational semantic parsing datasets, ATIS (Hemphill et al., 1990; Dahl et al., 1994), SPARC (Yu et al., 2019b), and COSQL (Yu et al., 2019a), show that our approach is able to maintain performance over long-range interactions, and several discourse related phenomena to a large extent.

Semantic Parsing for Conversational KGQA For tasks like conversational knowledge graph question answering (KGQA), it is not feasible to encode the entire external context (in this case entire KG) within the model. In Chapter 4, we introduce SPICE, a conversational semantic parsing dataset over knowledge graphs. SPICE is annotated with SPARQL queries which are executable on a real Wikidata engine and requires handling complex questions, types, relations, and entity linking on a large scale. Moreover, it showcases multiple linguistic phenomena such as coreference and ellipsis. We present two approaches that build on pretrained language models and provide detailed analysis, stratifying performance by question type and linguistic phenomena.

Dynamic Context Graphs In Chapter 5, we address the limitations of the methods proposed in Chapter 4. Due to the size of the external context, both of these make assumptions to reduce computational cost, e.g. by truncating the linearized subgraphs or encoding a smaller snapshot of the Wikidata ontology, sacrificing generalizability and the ability to predict unseen types or relations. To address these limitations we present dynamic graph for modeling external context. Our dynamic graphs are query-dependent and represent relevant context efficiently compared to linearized representations. We also show that the graph structure itself is crucial and outperforms simply converting the subgraph into a textual string.

Reasoning over Context Graphs with Large Language Models In Chapter 6, we present a method to integrate external graph context with large language models (LLMs). Our approach maintains a memory module to track and update past evidence, which is further incorporated in the context graph. Our results on the CONVMIX (Christmann et al., 2022b) dataset reveal that for the task of conversational question answering over multiple textual modalities such as tables, graphs, text, and infoboxes we outperform previous methods and LLMs without access to structured information. The graph enhances the LLM's ability to reason over multiple modalities, while the memory module provides robustness against noise and retrieval errors. Our proposed re-ranking based memory management further reduces computational requirements by storing only relevant facts, and forgetting those deemed unnecessary.

Part of the content covered in the thesis has been previously published in the Transactions of the ACL (Jain and Lapata (2021); Chapter 3), in EACL 2023 (Perez-Beltrachini et al. (2023a); Chapter 4), in EMNLP 2023 (Jain and Lapata (2023); Chapter 5), and is under revision at ACL 2025 (Jain and Lapata (2024); Chapter 6).

Chapter 2

Background

In this chapter we formulate the problem of conversational language understanding in the context of external knowledge. In Section 2.1, we classify and define various types of context we consider and define the problem in Section 2.2. We then describe the datasets involved in Section 2.4 and the evaluation metrics in Section 2.5. We also outline the necessary background and assumptions for the models proposed in subsequent chapters.

2.1 Conversational Language Understanding in Context

In this thesis, we define conversational language understanding as the ability of natural language interface systems to accurately interpret and respond to user queries in a conversational manner. We particularly focus on tasks that require access to external knowledge for effective comprehension and response. Such a task involves not only grasping the literal meaning of a user's utterance but also inferring its implicit intent and maintaining context throughout the interaction (Dijk, 2008). In this thesis, we assume that the relevant context is supplied before the interpretation starts and is not derived concurrently (Mazzone, 2015). We consider discourse and external contexts, and argue that effective natural language interface systems must incorporate both contexts to accurately respond to user inquiries.

2.1.1 Discourse Context

We define discourse context as the information that has been previously mentioned within the ongoing interaction (Dijk, 2008). Incorporating discourse context requires

co-reference resolution, ellipsis handling, and managing topic shifts (Chai and Jin, 2004; Sun and Chai, 2007a).

Ellipsis refers to the omission of information from an utterance that can be recovered from the context. In the interaction below, Q2 and Q3 exemplify nominal ellipsis, the NP *all flights from Long Beach to Memphis* is elided. Q4 is an example of coreference, *they* refers to the answer of Q3. However, it can also be recovered by considering all previous utterances (i.e., Where do they [flights from Long Beach to Memphis; any day] stop).

Q1: Please give me all flights from Long Beach to Memphis
Q2: What about 1993 June thirtieth
Q3: How about any day
Q4: Where do they stop

Focus-shift typically occurs when the interlocutor moves the conversation from one topic (entity) to another. Successfully handling such shifts is crucial for processing language in context in order to provide accurate responses. The focus of attention in each utterance is underlined in the example below, which in this case is operationalized as the most salient entity (e.g., city) within the utterance (Grosz et al., 1995). In the example below, the focus shifts from *flights* in Q2 to *cities* in Q3.

Q1: What flights are provided by American airlines
Q2: What flights are provided by Delta airlines
Q3: Which cities are serviced by both American and Delta airlines

2.1.2 External Context

External context involves information that is not explicitly stated within the conversation but is available as world knowledge. We define *external* context specifically as the kind of global world knowledge represented in knowledge based or knowledge graphs, excluding other types such as situational context (Dijk, 2008). Reasoning over external knowledge sources is vital for providing an accurate response. For example, in responding to the query:

Q: Please give me all flights from Long Beach to Memphis.

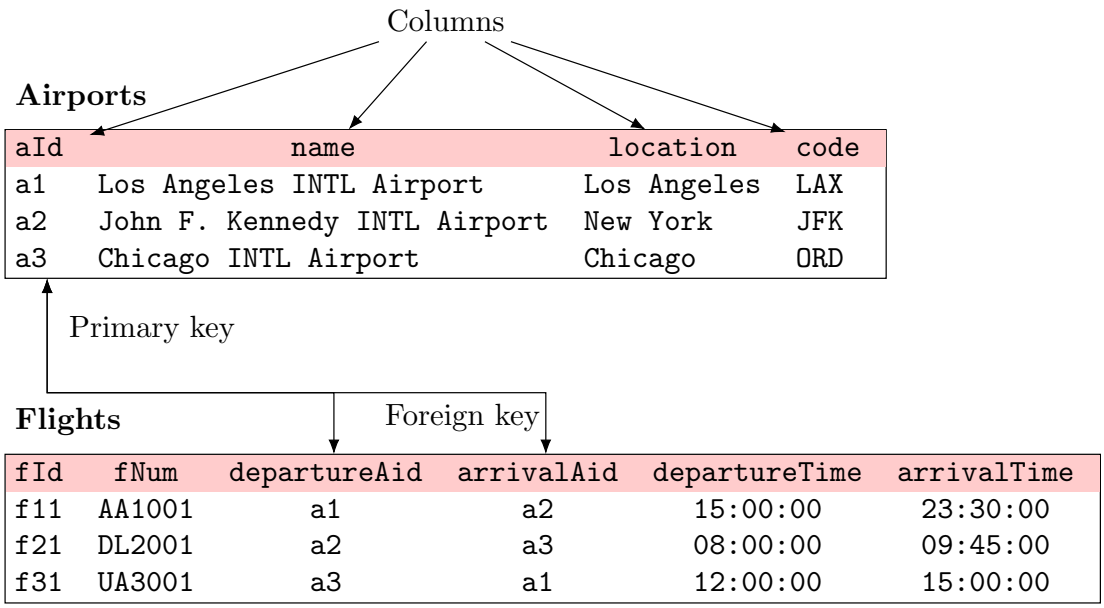

Figure 2.1: Example of a relational database schema. The figure shows multiple tables, each represented by a box. The name of the table is displayed at the top of the box (Airports and Flights). Inside each box, the columns names are shown in the first row followed by rows containing values. The line connecting aId to departureAid arrivalAid and indicates a foreign key constraint.

The system needs access to external information, in this case, a database containing flight details, and needs to reason over information available about available flights. Specifically, it should list flights departing from Long Beach and arriving at Memphis. Such explicit reasoning and real-time access to the knowledge source is necessary for two primary reasons. First, this approach eliminates the need for the model to memorize all possible facts. Second, information is constantly updated; for instance, flight availability changes over time, which can render current information obsolete.

In this thesis, we consider three different types of external context that cover a variety of use-cases. Firstly, we utilize relational databases accessible via SQL (Chamberlin and Boyce, 1974) queries (Chapter 3). Secondly, the Wikidata (Vrandečić and Krötzsch, 2014) knowledge graph (KG), which can be accessed using a KG engine via a query language such as SPARQL (Chapter 4 and 5), or directly through a web interface. Finally, we explore an open-domain setting, where user queries require reasoning over Wikipedia text, tables, and infoboxes *and* Wikidata (Chapter 6).

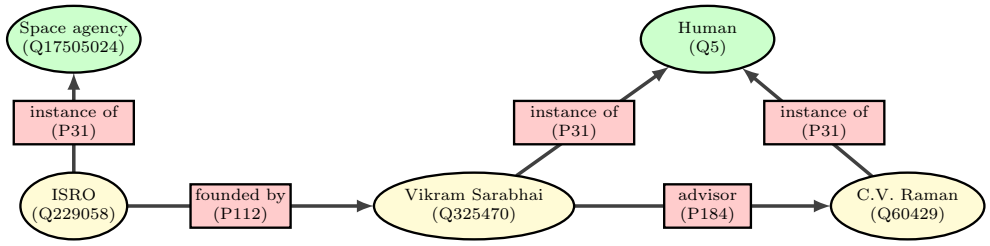

Figure 2.2: Example of a Wikidata subgraph. Each ◯ colored node represents an entity and ◯ a class, and relationships between nodes in the graph are depicted using ▭ colored rectangular boxes. Direction of the arrows shows the directionality of the relation. Wikidata identifiers for each entity and relationship are displayed within brackets.

2.1.2.1 Relational Databases

Relational databases (Codd, 1970) are optimized for storing large amount of data in tabular form. A database schema defines the structure of the tables, columns within each table and how tables and columns relate to each other. Figure 2.1 shows an example database schema that stores details of different airports and the operational flights. The example database consists of two tables: Airports and Flights. Each table lists its columns along with their respective data types. For instance, Airports table includes columns airport id aId, name, code and location. Tables are linked through the *foreign key* constraints that enforces referential integrity. This means that certain columns in one table reference columns in another table to ensure consistent data relationships. For example, in Figure 2.1, departureAid and arrivalAid of Flights table refer to aId column in the Airports table.

2.1.2.2 Knowledge Graph

Precise definition of what constitutes a knowledge graph remains contentious (Ehrlinger and Wöß, 2016; Bonatti et al., 2019). The core idea is to represent data in the form of a graph. Most well-known knowledge graphs (KG), such as DBpedia (Lehmann et al., 2015), Wikidata (Vrandečić and Krötzsch, 2014), and YAGO (Hoffart et al., 2011) span various domains. These graphs are either derived from Wikipedia (as with DBpedia and YAGO) or constructed by communities of volunteers (such as Wikidata). Apart from such general purpose knowledge graphs, many companies build special purpose enterprise knowledge graphs (Noy et al., 2019) for varied use cases, including, search (Singhal, 2012; Shrivastava, 2017), product (Krishnan, 2018), and finance (Tobin, 2017; Elham-

madi et al., 2020). In this thesis, we define a Knowledge Graph (KG) as a data structure designed to organize information about the real world by using nodes to represent entities and edges to depict the relationships between these entities.

We utilize the Wikidata knowledge graph, which is one of the largest, continuously evolving, and open-access knowledge graphs available (Haller et al., 2022). Wikidata adheres to a specific ontology[1] which outlines the overall structure of the graph. Our work involves the use of entities, properties, and classes. In Wikidata, an Entity is a unique item with a distinct identifier, a Class defines a category of entities (often represented as a subclass of the main concept), and a Property describes a relationship or attribute between entities.

Figure 2.2 displays an example subgraph from Wikidata. Each KG item has a unique identifier. For instance, the entity Vikram Sarabhai is assigned the identifier Q325470, while the class Human is designated the identifier Q5. Like entities and classes, properties in Wikidata also have unique identifiers that begin with 'P', contrasting with 'Q', which is used for entities and classes. For example, in the subgraph depicted in Figure 2.2, P112 corresponds to the property founded by. Additionally, it is important to note that the schema of any given entity is dynamically instantiated. This means that specific attributes, such as advisor, may not be present in all instances of an entities belonging to Human class. As we will discuss further in Chapter 4, this requires dynamically grounding the current user utterance to the corresponding subgraph, which is challenging than working with the static schema of a relational database. However, this flexibility also allows for the addition of new knowledge (even if incomplete) to the existing graph.

2.2 Problem Formulation

We operate within a conversational question-answering (QA) setting. We define an interaction I as a sequence of turns consisting of user utterances and system responses. Specifically, an interaction is represented as $I = \langle X_1, A_1, X_2, A_2, \ldots, X_T, A_T \rangle$, where each X_t is a user utterance and each A_t is the corresponding response from the system at turn t. To successfully respond to X_t, the system must consider the previous QA pairs – the *discourse context* up to turn t – denoted as $I[:t-1]$, and the task-dependent *external context* C. Each user utterance X_t is presented as a sequence of natural language tokens $\langle x_1, x_2, \ldots, x_{|X_t|} \rangle$, where $|X_t|$ represents the length of the sequence, and each x_i $(i \in [1, |X_t|])$ is a single natural language token. Similarly, the response at time t is

[1] https://www.wikidata.org/wiki/Wikidata:WikiProject_Ontology

composed of the sequence of tokens $\langle a_1, a_2, \ldots, a_{|A_t|} \rangle$. The computational approach to deriving answer A_t can vary based on the external context and the task itself. It may involve initially parsing X_t into an executable semantic interpretation Z_t, then executing Z_t on a query engine to fetch the response, or directly generating the response based on the query and contextual information. In the example below, based on the database presented in Figure 2.1 as context C, we show a user utterance X_1, the SQL query Z_1, and the answer A_1 obtained from executing Z_1. The user query requires reasoning over both Airports and Flights tables, as well as the foreign-key relationship between them, implemented using a SQL JOIN operator.

X_1: Which airports have flights departing after noon?
Z_1: SELECT DISTINCT a.name FROM Airports a JOIN Flights f ON a.aid = f.departureAid
WHERE f.departureTime > '12:00:00';
A_1: Los Angeles INTL Airport

In some scenarios, it is possible to answer the question directly. In the example below, based on the evidence retrieved from the subgraph presented in Figure 2.2 as external context C, it is possible to answer the user question directly.

X_1: Who is the advisor of Vikram Sarabhai?
Evidence: < Vikram Sarabhai, advisor, C.V. Raman >
A_1: C.V. Raman

In Chapter 6, we will discuss how such evidence can be retrieved based on the user query. We will also present methods to combine evidence from different sources, such as tables and knowledge graphs.

2.2.1 Semantic Parsing for Conversational Question Answering

Semantic parsing (Dong and Lapata 2016; Jia and Liang 2016; Iyer et al. 2017; Agrawal et al. 2019; Wang et al. 2020; Zhong et al. 2018; Yu et al. 2018, *inter alia*) based methods for conversational question answering (Shen et al., 2019; Kacupaj et al., 2021) convert a user's utterance X_t into a formal representation Z_t. Z_t is then executed on an external query engine (for example, a relational database or a knowledge graph engine) to obtain answer A_t. Hence, the primary objective of semantic parsing is to predict a formal representation that when executed returns the correct answer. So the question

answering model $\mathcal{M}_{sp}$ based on semantic parsing is formulated as,

$$Z_t = \mathcal{M}_{sp}\left(\mathrm{I}[:t-1], X_t, \mathrm{C}; \Theta\right) \tag{2.1}$$

$$A_t = \mathrm{Execute}\left(Z_t, \mathrm{C}\right) \tag{2.2}$$

Where Θ are the parameters of the model and I[:t - 1] is represents the history of the user utterance $X_{<t}$, semantic parse $Z_{<t}$, and answers $A_{<t}$. External context is denoted using C and operator $\mathrm{Execute}$ represents the query engine that executes parse Z_t to obtain A_t.

In this thesis, we will train the semantic parser $\mathcal{M}_{sp}$ using a supervised approach. Specifically, we assume our dataset includes both the correct answers A_t and the corresponding reference parses Z_t for each utterance X_t during training.

Formal Representation A critical element in the development of a semantic parsing model is choosing a machine-readable formal representation. Compared to code generation (Jiang et al., 2024) where we have high-level specifications (Lee et al., 2021), in this thesis, we are interested in meaning representations that allow us to capture the precise meaning of user query. Meaning representations dictate the structure and articulation of meaning. Various representations have been proposed, including λ-calculus, λ-dependency-based compositional semantics (λ-DCS; Liang et al. (2013); Liang (2013)), AMR (Banarescu et al., 2013), and structured query languages like SQL (Chamberlin and Boyce, 1974). We use SQL and SPARQL, as they can be executed on real-world data engines.

Entity Linking Semantic parsing with knowledge graph context requires entity linking. For example, mapping Vikram Sarabhai to Q325470, which involves identifying mentions (Ma and Hovy, 2016; Devlin et al., 2019) of entities within the user query and mapping them to their corresponding unique identifiers in the knowledge graph. It allows the semantic parser to ground the natural language input in the structured knowledge of the graph. In this thesis, we use off the shelf model such as AllenNLP [2] and CLOCQ [3] for named entity recognition and linking. We present task dependent details in Chapter 4 5 and 6.

[2]https://docs.allennlp.org/main/
[3]https://clocq.mpi-inf.mpg.de/documentation

Query Engine Depending on the task and the external knowledge context, different query execution engines are used in this thesis. For Wikidata KG, we use Blazegraph[4] which facilitates the local deployment of a triple store and the efficient execution of SPARQL queries. For relational databases, we utilize the MySQL[5] and SQLite (Owens, 2006) database engines.

2.2.2 Direct Response Generation

The semantic parsing based approach discussed in Section 2.2.1 requires labeled formal queries. These are costly to annotate and require expert annotators. Moreover, for tasks involving KG question answering, the underlying assumption is that the KG is complete with no missing links, which is not the case with real-world KGs. Given user utterance X_t, a direct response generation model $\mathcal{M}_{DR}$ learns to directly generate a response A_t. The task is formulated as follows:

$$A_t = \mathcal{M}_{DR}\left(\mathsf{I}[:t-1], X_t, \mathsf{C}; \Theta\right) \tag{2.3}$$

Where $\mathsf{I}[:t-1]$ represents the user utterance $X_{<t}$ and answer $A_{<t}$ history. C denotes external context, while Θ represents the model parameters. Unlike semantic parsing based model in Equation 2.1, which learns to generate parse Z_t, model $\mathcal{M}_{DR}$ directly generates response A_t. Furthermore, this formulation allows perform question answering over heterogeneous resources simultaneously. In Chapter 6, we will use this formulation for conversational question answering over multiple textual modalities.

2.3 Information Retrieval for Conversational Question Answering

Question answering over a large external context, such as, knowledge graph or open domain web-scale data, uses information retrieval methods (Gao et al., 2019b) to gather evidence which allows efficient reasoning to answer the user query. In direct response generation based approach, discussed in Section 2.2.2, external context C can be scaled down by using retrieval techniques to retrieve small text snippets containing the answer. These are then contextually processed with the user query to generate the final answer (Izacard et al., 2024; Khandelwal et al., 2020; Guu et al., 2020). A popular

[4] https://blazegraph.com/
[5] https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html

	ATIS	SPARC	COSQL	SPICE	CONVMIX
No. Instances	1,658	4,298	3,007	197K	2800 (3000*)
Avg. No. turns	7.0	3.0	5.2	9.5	5 (10*)
Domain	Single	CD	CD	KG	Wikipedia, KG
Logical form	SQL	SQL	SQL	SPARQL	–
Database type	DB	DB	DB	KG	–
No. Databases	1	200	200	–	–
No. Tables	27	1020	1020	–	–
Unanswerable Ques.	✗	✗	✓	✓	✗
Executable	✓	✓	✓	✓	✗

Table 2.1: Datasets used in this thesis. CD indicates cross-domain where the databases used during test are not seen during training. DB is relational database and KG is Wikidata knowledge graph. *No.* is the abbreviations for *number of*. * CONVMIX contains a separate out-of-domain test set (CONVMIX-10T, discussed further in Chapter 6) with an average interaction length of 10 compared to 5 for rest of the dataset.

approach is a two-stage process of retrieval followed by ranking. Retrieval focuses on identifying a large set of potentially relevant items, while ranking refines this set by assigning scores and ordering them to present the most relevant items first. String based/lexical retrievers, such as using BM25 (Robertson and Zaragoza, 2009; Robertson and Jones, 1977), are commonly used which are fast and scalable but cannot process contextual information effectively. To overcome the noise in retrieval, dense rankers are used that scores the relevancy of all candidates for the given search query (Reimers and Gurevych, 2019; Chen et al., 2020a). We use information retrieval based methods in different places in this thesis and the relevent details are presented within the chapters. In Chapter 5, we use elastic search for entity linking. In Chapter 6, we use BM25 for evidence retrieval based on the user query and SBERT (Reimers and Gurevych, 2019) for ranking.

2.4 Datasets

In this thesis, we use ATIS (Hemphill et al., 1990; Dahl et al., 1994), SPICE (Perez-Beltrachini et al., 2023a), SPARC (Yu et al., 2019b), and COSQL (Yu et al., 2019a) dataset. All datasets are conversational in nature and primarily vary based on the

external context and task requirements. In Table 2.1 we summarizes the statistics for each of them.

2.4.1 ATIS

The ATIS (Hemphill et al., 1990; Dahl et al., 1994) dataset, curated following the wizard-of-oz technique, involves a human operator – known as the "wizard" – who simulates an automated system. Users, under the impression they are interacting with a machine, submit spoken inquiries about travel information. These inquiries are manually processed by the wizard, who has access to relevant databases, to simulate a responsive automated system.

In this thesis, we use the ATIS split created by Suhr et al. (2018), that ensures no interaction scenarios overlap between the train and test. The dataset contains interactions in the form of multi-turn user utterance and responses in the form of SQL. As shown in Table 2.1, the dataset comprises 1,658 multi-turn interactions, each with an average length of 7 turns. These interactions involve flight booking scenarios, expressed through user utterances and corresponding SQL queries. The underlying relational database used in these scenarios is fixed, containing 27 tables and 162K entries. Table 2.2 shows example input utterances and corresponding SQL parse.

2.4.2 SPARC and COSQL

Contrary to the ATIS (Suhr et al., 2018) dataset, which employs a fixed database, both SPARC (Yu et al., 2019b) and COSQL (Yu et al., 2019a) assume a cross-domain environment. In this context, 'cross-domain' implies that the databases used during testing are not seen during training. Table 2.3 shows an example interaction for both the datasets. Each user utterance in SPARC (Yu et al., 2019b) and COSQL (Yu et al., 2019a) is mapped to an SQL query. COSQL, unlike SPARC, contains a significant number of clarification questions that cannot be unambiguously mapped to SQL queries.

As can be seen in example presented in Table 2.2, SQL queries in ATIS predominantly feature joins and nested subqueries. However, several common SQL components ,such as ORDER BY, HAVING, and SET, are entirely absent, while others like GROUP BY and AGGREGATE functions are rarely used. In contrast, the SPARC and COSQL (see Table 2.3) datasets encompass all primary SQL components, thereby offering a more comprehensive range of SQL query capabilities (Yu et al., 2019b).

ATIS
1. What Continental flights go from Chicago to Seattle before 10 am in morning 1993 February twenty sixth SELECT DISTINCT flight.flight_id FROM flight WHERE (flight.airline_code = 'CO' AND (flight.from_airport IN (SELECT airport_service.airport_code FROM airport_service WHERE airport_service.city_code IN (SELECT city.city_code FROM city WHERE city.city_name = 'CHICAGO')) AND (flight.to_airport IN (SELECT airport_service.airport_code FROM airport_service WHERE airport_service.city_code IN (SELECT city.city_code FROM city WHERE city.city_name = 'SEATTLE')) AND (flight.departure_time < 1000))))
2. Continental flights before noon that have a meal SELECT DISTINCT flight.flight_id FROM flight WHERE (flight.airline_code = 'CO' AND (flight.from_airport IN (SELECT airport_service.airport_code FROM airport_service WHERE airport_service.city_code IN (SELECT city.city_code FROM city WHERE city.city_name = 'CHICAGO')) AND (flight.to_airport IN (SELECT airport_service.airport_code FROM airport_service WHERE airport_service.city_code IN (SELECT city.city_code FROM city WHERE city.city_name = 'SEATTLE')) AND (flight.departure_time < 1200 AND flight.meal_code IN (SELECT food_service.meal_code FROM food_service WHERE 1 = 1)))))
3. Continental flights before 2 pm SELECT DISTINCT flight.flight_id FROM flight WHERE (flight.airline_code = 'CO' AND (flight.from_airport IN (SELECT airport_service.airport_code FROM airport_service WHERE airport_service.city_code IN (SELECT city.city_code FROM city WHERE city.city_name = 'CHICAGO')) AND (flight.to_airport IN (SELECT airport_service.airport_code FROM airport_service WHERE airport_service.city_code IN (SELECT city.city_code FROM city WHERE city.city_name = 'SEATTLE')) AND (flight.departure_time < 1400 AND flight.meal_code IN (SELECT food_service.meal_code FROM food_service WHERE 1 = 1)))))
4. On 1993 February twenty seventh SELECT DISTINCT flight.flight_id FROM flight WHERE (flight.airline_code = 'CO' AND (flight.from_airport IN (SELECT airport_service.airport_code FROM airport_service WHERE airport_service.city_code IN (SELECT city.city_code FROM city WHERE city.city_name = 'CHICAGO')) AND (flight.to_airport IN (SELECT airport_service.airport_code FROM airport_service WHERE airport_service.city_code IN (SELECT city.city_code FROM city WHERE city.city_name = 'SEATTLE')) AND (flight.departure_time < 1400 AND (flight.flight_days IN (SELECT days.days_code FROM days WHERE days.day_name IN (SELECT date_day.day_name FROM date_day WHERE date_day.year = 1993 AND date_day.month_number = 2 AND date_day.day_number = 27)) AND flight.meal_code IN (SELECT food_service.meal_code FROM food_service WHERE 1 = 1))))))

Table 2.2: Example interaction from ATIS (Suhr et al., 2018) dataset. The dataset uses relational database as external context. Each interaction turn consists of utterance and corresponding SQL parse. In this case, the *answer* obtained from executing each SQL query consists of numeric flight IDs.

Table 2.1 presents statistics for SPARC (Yu et al., 2019b) and COSQL (Yu et al., 2019a). Both datasets feature 200 different databases, encompassing a total of 1,020 tables. These characteristics make the datasets valuable benchmarks for measuring cross-domain generalization capabilities. They require models to adapt not only to a new domain but also to different database schema designs. However, compared to other datasets, the average interaction length is limited to 3 for SPARC and 5.2 for COSQL.

2.4.3 SPICE

ATIS (Suhr et al., 2018), SPARC (Yu et al., 2019b), and COSQL (Yu et al., 2019a) are specifically designed for relational databases, which are generally limited to specific domains. In contrast, large knowledge graphs like Wikidata can handle queries across a wide range of topics. To address this limitation, Perez-Beltrachini et al. (2023a) created the SPICE dataset.

Table 2.4 shows an example of an interaction from the SPICE dataset. It is a large-scale conversational question answering dataset over Wikidata. The dataset comprises of pairs of user questions in natural language and their corresponding SPARQL queries. The answers to these questions are obtained by executing the queries on the Wikidata KG engine. SPICE is derived by semi-automatically adding SPARQL queries to the existing CSQA (Saha et al., 2018a) benchmark, which was originally proposed for retrieval-based conversational question answering.

Table 2.1 shows the statistics of the SPICE dataset. It is a large-scale dataset containing $\sim 197,000$ conversations with an average interaction length of 9.5 turns. This dataset necessitates reasoning over Wikidata KG, which comprises millions of entities and thousands of relation types. Such complexity makes it ideal for testing the ability of machine learning models to handle extensive external contexts within conversational settings.

2.4.4 ConvMix

Most conversational question answering systems operate over single information source, for example, a knowledge base (Perez-Beltrachini et al., 2023a; Saha et al., 2018a), or a text corpus (Chen et al., 2020b; Huang et al., 2018; Qiu et al., 2021; Qu et al., 2019b,a), or tables (Iyyer et al., 2017a; Yu et al., 2019b; Jain and Lapata, 2021; Mueller et al., 2019a). In a real-world applications, it is reasonable to expect these information sources to be complementary. For example in Table 2.5, while Wikipedia text and

SPARC
Database: Concerts
1. Show the names of stadiums. SELECT name from stadium Stark's Park; Somerset Park; Bayview Stadium; Hampden Park; Forthbank Stadium; Gayfield Park; Recreation Park; Balmoor
2. For each of them, also show the number of concerts. SELECT T2.name , count(*) FROM concert AS T1 JOIN stadium AS T2 ON T1.stadium_id = T2.stadium_id GROUP BY T1.stadium_id (Stark's Park, 1) ; (Glebe Park, 1); (Somerset Park, 2); (Recreation Park, 1); (Balmoor, 1)
CoSQL
Database: Student Transcripts Tracking
1. How many courses in this table? — Do you mean the number of the courses with different course ids? — Yes. SELECT count (distinct course_id) from courses 15
2. What's the name of the course with least number of students of enrollments? SELECT T1.course_name FROM Courses AS T1 JOIN Student_Enrolment_Courses AS T2 ON T1.course_id = T2.course_id GROUP BY T1.course_name ORDER BY count(*) LIMIT 1 AI

Table 2.3: Example interaction from SPARC (Yu et al., 2019b) (top), and COSQL (Yu et al., 2019a) (bottom) dataset. Both the dataset are cross-domain and uses relational database as *external* context. Each interaction consists of user utterance, SQL and the corresponding answer.

Wikidata can be used to answer *What is the release date of album Kid A?*, but Wikipedia tables cannot. Conversely, Wikipedia tables can determine *What is Kid A's ranking on Rolling Stone in 2009?* but this information is not available in Wikipedia text or Wikidata. CONVMIX (Christmann et al., 2022b) is a benchmark designed to evaluate systems that can answer questions by combining information from various sources, including text, tables, infoboxes, and the Wikidata knowledge graph.

Table 2.1 presents statistics for the CONVMIX dataset. It comprises 2,800 conversations, each with five turns. To evaluate out-of-domain performance in terms of conversation length, a separate test set, ConvMix-10T is included. Which contains 200 conversations with ten turns each.

SPICE
1. Who starred in Mathias Kneissl ? SELECT ?x WHERE { wd:Q3298576 wdt:P161 ?x . ?x wdt:P31 wd:Q502895 . } Rainer Werner Fassbinder, Volker Schlöndorff, Hanna Schygulla
2. Who was the director of that work of art ? SELECT ?x WHERE { wd:Q3298576 wdt:P57 ?x . ?x wdt:P31 wd:Q502895 . } Reinhard Hauff
3. Does Dubashi have that person as actor ? ASK { wd:Q76025 wdt:P161 wd:Q24807818 . } No
4. Which works of art are Rainer Werner Fassbinder or Laura Esquivel a screenwriter of ? SELECT ?x WHERE { { ?x wdt:P58 wd:Q44426 . ?x wdt:P31 wd:Q838948 . } UNION { ?x wdt:P58 wd:Q230586 . ?x wdt:P31 wd:Q838948 . } } The American Soldier, Lili Marleen, Love Is Colder Than Death ...
Q3298576: Mathias Kneissl, Q76025: Reinhard Hauff, Q24807818: Dubashi, Q44426: Rainer Werner Fassbinder Q230586: Laura Esquivel, Q838948: work of art, Q502895: common name, P161: cast member, P31: instance of, P57: director, P58: screenwriter

Table 2.4: Example interaction from SPICE dataset with utterances, corresponding SPARQL queries, and answers returned after executing the queries on the Wikidata graph engine. The bottom block shows the KG elements (i.e., graph nodes) involved in this interaction.

CONVMIX
1. What is the release date of album Kid A? 2 October 2000
2. Fact Rank? 7
3. Ranking on Rolling Stone in 2009? 1
4. Editor? Noah Shachtman
5. Category? Popular culture

Table 2.5: Example interaction from CONVMIX dataset that uses Wikipedia text, tables and Infobox, and Wikidata knowledge graph as *external* context. Each turn in the example interaction consists of user question and the system response.

2.5 Evaluation Metrics

We discuss the evaluation metrics employed in this thesis. These metrics fall into two broad categories: those that necessitate the use of an execution engine for evaluation, and those that involve string-based matching of either the semantic parse or the generated answer.

2.5.1 Exact Match

The exact match metric measures the accuracy of the system's output against a gold standard answer. We define an exact match for each comparison between the system *output* and the expected correct string, as follows:

$$\mathbb{1}_{\text{EM}}(\text{output}, \text{expected}) = \begin{cases} 1 & \text{if output} = \text{expected}, \\ 0 & \text{otherwise.} \end{cases} \tag{2.4}$$

We then calculate the overall exact match accuracy using the formula:

$$\text{Exact Match Accuracy} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\text{EM}}(\text{output}_{\text{i}}, \text{expected}_{\text{i}}) \tag{2.5}$$

Where n is the total number of instances. In the context of semantic parsing tasks in this thesis, we consider the generated semantic parses (SQL or SPARQL) as single strings, and match these against corresponding gold standard parses.

2.5.2 Denotation Accuracy

In semantic parsing tasks, a user utterance may have multiple valid interpretations. Comparing these interpretations through simple string matching can be misleading. Consider the following example where both SQL statements S1 and S2 adequately satisfy the query Q:

> Q: Show me the names of employees who earn more than 3000.
> S1: SELECT name FROM employees WHERE salary $>$ 3000
> S2: SELECT e.name FROM employees e WHERE salary $>$ 3000

Both S1 and S2 SQL queries are structurally different but semantically equivalent. That means, even a minor difference in the queries will incorrectly penalize the model. To compute denotation accuracy, we first execute the output and gold queries, and then

perform a match on the results. Specifically, denotation-based accuracy for generated query $\hat{Z}_t$ and gold query Z is computed as:

$$\text{Denotation Accuracy} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{\text{EM}}(\text{Execute}(\hat{Z}_t), \text{Execute}(Z_t)) \tag{2.6}$$

Here, $\mathbb{1}_{\text{EM}}$ denotes the Exact Match indicator function, as defined in Equation 2.4. Execute represents the query engine that executes parse Z_t to obtain the answer.

2.5.3 F1 Score

The F1 (Sasaki et al., 2007) score is particularly useful for tasks that return a set of items as answers, such as question answering over a Knowledge Graph (KG). It calculates the harmonic mean of precision and recall. Let Y be the set of gold items and $\hat{Y}$ is predicted set. The F1 score can be expressed as:

$$\text{F1} = 2 \cdot \frac{\text{P} \cdot \text{R}}{\text{P} + \text{R}} \tag{2.7}$$

where P, or precision, is the proportion of predicted answers that are correct, defined as:

$$P = \frac{|\hat{Y} \cap Y|}{|\hat{Y}|}$$

and R, or recall, is the proportion of correct answers that were predicted:

$$R = \frac{|\hat{Y} \cap Y|}{|Y|}$$

For semantic parsing based methods for question answering over knowledge graph, Y is obtained by executing the output parse:

$$Y = \text{Execute}(Z_t)$$

2.6 Base Model Architectures

In this section, we describe the base models used in this thesis. In Section 2.6.1, we will discuss Long Short-Term Memory (LSTM; Hochreiter and Schmidhuber (1997)) networks, which we utilize in Chapter 3. We will introduce an encoder-decoder architecture (Cho et al., 2014) in Section 2.6.2, which forms the basis of the models presented in Chapters 3, 4, and 5. Next, we will describe transformer-based models (Vaswani et al., 2017a) that have been shown to outperform variants of recurrent neural networks,

as they facilitate parallel and large-scale training (Vaswani et al., 2017a). Finally, Section 2.6.4 provides a brief introduction to graph neural networks (GNNs; Gori et al. (2005a); Scarselli et al. (2008)). We use GNNs in Chapters 5 and 6 to encode the structural information present in the *external* context.

2.6.1 Long Short-Term Memory Networks (LSTMs)

Long Short-Term Memory (LSTM; Hochreiter and Schmidhuber (1997)) are a specialized form of recurrent neural networks designed to address the vanishing gradient problem inherent in traditional RNNs. This is accomplished through the use of LSTM cells, which consists of several gates: input, forget, and output, which regulate the flow of information throughout the cell. These gates play a critical role in determining whether information is retained or discarded.

Given input utterance $X = \langle x_1, x_2 \cdots x_{|X|} \rangle$, an LSTM cell process token $x_t \in \mathbb{R}^d$ at time $t \in [1, |X|]$ as,

$$f_t = \sigma(W_{if} x_t + W_{hf} h_{t-1} + b_f) \quad \text{(f = Forget Gate)} \tag{2.8}$$

$$i_t = \sigma(W_{ii} x_t + W_{hi} h_{t-1} + b_i) \quad \text{(i = Input Gate)} \tag{2.9}$$

$$o_t = \sigma(W_{io} x_t + W_{ho} h_{t-1} + b_o) \quad \text{(o = Output Gate)} \tag{2.10}$$

The cell updates are governed by:

$$c_t = f_t \odot c_{t-1} + i_t \odot tanh(W_{ig} x_t + W_{hg} h_{t-1} + b_g) \tag{2.11}$$

$$h_t = o_t \odot \tanh(c_t) \tag{2.12}$$

Where $\odot$ is the element-wise product and σ is the sigmoid layer. Input weight matrices W_{i*} are of dimensions $W_{i*} \in \mathbb{R}^{hxd}$ and weight matrices corresponding to hidden representations W_{h*} are of dimensions $W_{h*} \in \mathbb{R}^{hxd}$. Specifically, W_{if}, W_{ii}, and W_{io} are the input weights; W_{hf}, W_{hi}, and W_{ho} are the hidden representation weights; and b_f, b_i, and b_o are the biases for forget (f_t), input (i_t) and output (o_t) gates. c_t in Equation 2.11 is the updated LSTM cell state that is calculated by deciding how much to forget from previous state c_{t-1} and how much new information should be added based on input gate i_t. Weights W_{ig} and W_{hg} transforms the current input x_t, previous hidden state h_{t-1} to calculate new information, with b_g as bias parameter. $tanh$ is the hyperbolic tangent activation function that maps input values to a range between -1 and 1. Being zero-centered it help backpropagate through multi-layer neural networks (Datta, 2020; Karlik and Olgac, 2011; Neal, 1992). h_t is the output hidden representation. Hidden representation at $h_{|X|}$ is used as final sequence representation.

2.6.2 Encoder-Decoder

The encoder-decoder architecture (Cho et al., 2014) is a general framework for sequence-to-sequence tasks, consisting of an *encoder* and a *decoder*. The encoder processes the input sequence $X = \{x_1, x_2, \ldots, x_{|X|}\}$, where $x_t \in \mathbb{R}^d$, and produces a hidden representation $H = \{h_1, h_2, \ldots, h_{|H|}\}$, where $h_t \in \mathbb{R}^k$. H can be either a fixed-length vector with $|H| = 1$ (e.g., in RNNs, see Section 2.6.1) or a sequence of vectors (e.g., in Transformers, see Section 2.6.3). The decoder generates the output sequence $Y = \{y_1, y_2, \ldots, y_{|Y|}\}$, where $y_t \in \mathbb{R}^m$, conditioned on H, often using attention mechanisms (Bahdanau et al., 2015a).

Formally, the encoder is a function $\mathrm{Encoder}$ that maps the input sequence to a hidden representation:

$$H = \mathrm{Encoder}(X)$$

The decoder is a function $\mathrm{Decoder}$ that generates the predicted output sequence $\hat{Y} = \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{|\hat{Y}|}\}$:

$$\hat{Y} = \mathrm{Decoder}(H)$$

2.6.3 Transformer

The fundamental building block of a transformer network (Vaswani et al., 2017a) is a transformer layer. It takes a set of input vectors $X = \langle x_1, x_2 \cdots x_{|X|} \rangle$, where X has dimensions $X \in \mathbb{R}^{N \times d}$ and $N = |X|$ is the number of tokens. The layer *transforms* them into a set of output vectors of the same dimensions $\bar{X} = \langle \bar{x}_1, \bar{x}_2 \cdots \bar{x}_{|X|} \rangle$, thus $\bar{X} \in \mathbb{R}^{N \times d}$.

$$\bar{X} = \mathrm{TransformerLayer}(X) \tag{2.13}$$

Each transformer layer computes contextual representations using multi-head self-attention. It enables the model to simultaneously attend to different subspaces of the input, enhancing its capacity to capture diverse relationships.

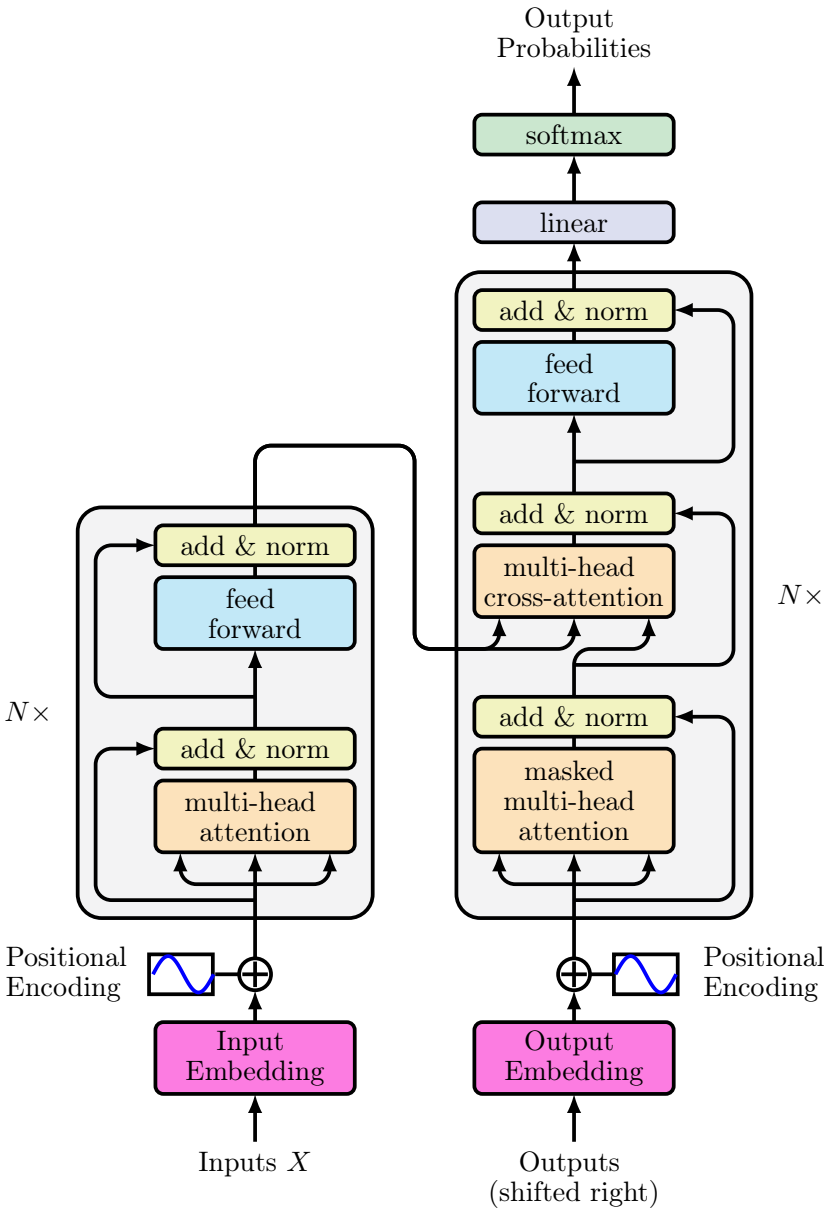

Figure 2.3: Illustration of Transformer Encoder-Decoder Architecture. The architecture consists of *input embedding* layer that converts the input sequence X into dense vector representations. Encoder is composed of several transformer layers, which processes the input embeddings. Each layer uses *multi-head self-attention* and *positional encoding* to capture token relationships and their order, followed by *add & norm*, which combines a residual connection with layer normalization to stabilize training. The Decoder which generates the output sequence, is also made up of several layers, employs a *masked multi-head self-attention* mechanism to prevent the influence of future token while incorporating *cross-attention* to reference the encoder's outputs. Nx indicates a block repeated n times.

In this thesis, by following (Vaswani et al., 2017a), we define self-attention as a scaled dot product:

$$\hat{X} = \text{Attention}(Q,K,V) = \text{SoftMax}\left[\frac{QK^T}{\sqrt{d'}}\right]V \tag{2.14}$$

In this formulation, Q, K and V represent the query, key, and value matrices, respectively, each derived through a linear transformation of the input X. Specifically,

$$Q = XW_q$$
$$K = XW_K$$
$$V = XW_V$$

Here, W_k, W_q, and W_v all have dimensions $\mathbb{R}^{N\times d}$. Given self-attention as defined in Equation 2.14, multi-head attention computes H independent heads, each with its own learnable parameters. Each head h, where $h \in [1\cdots H]$, is computed as follows:

$$\hat{X}_h = \text{Attention}(Q_h, K_h, V_h) \tag{2.15}$$
$$\hat{X}_H = \oplus[\hat{X}_1, \hat{X}_2 \cdots \hat{X}_\mathbf{h}]\, W_o \tag{2.16}$$

Where $\oplus$ denotes concatenation and W_o is the output transformation matrix. Equations 2.15 and 2.16 together define the multi-head self-attention mechanism. Each output matrix $\hat{X}_h$ of multi-head attention is of shape $\mathbb{R}^{N\times d}$, so the final matrix has dimensions $\hat{X}_h \in \mathbb{R}^{N\times Hd}$. The output matrix projects the concatenated output back to the original shape, i.e, $W_o \in \mathbb{R}^{Hd\times d}$.

$$L = \text{LayerNorm}(\hat{X}_H + X) \tag{2.17}$$
$$\bar{X} = \text{LayerNorm}(\text{FF}(L) + X) \tag{2.18}$$

Equation 2.18 describes the computation within a single TransformerLayer represented by Equation 2.13. Here, FF is a feed-forward network, and LayerNorm is the layer normalization (Ba et al., 2016) that is used along with residual connections (He et al., 2016) to improve training stability and efficiency.

Figure 2.3 shows the overall transformer architecture. A transformer encoder consists of N layers, each of a type TransformerLayer. The transformer decoder (Vaswani et al., 2017a) attends over encoded outputs using *cross-attention*. Cross-attention operates similarly to self-attention, except that the key and value vectors are taken from the final layer of the transformer encoder.

2.6.4 Graph Attention Networks

Neural networks have been adapted to leverage the structure and properties of graphs, resulting in a class of models known as Graph Neural Networks (GNNs; Gori et al. (2005a); Scarselli et al. (2008)). GNNs operate by learning representations of nodes, edges, and, potentially, entire graphs, which capture the connectivity patterns and feature information that are unique to each graph element.

In this thesis, we use graph neural networks (GNNs) to learn node representations by aggregating information from neighboring nodes. We assume a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ are nodes and edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. Each GNN layer l, takes as input node representations $\{h_i^{l-1} \mid i \in [1,n]\}$ corresponding to nodes $\mathcal{V}$ and edges $\mathcal{E}$. The output of each layer is an updated set of node representations $\{h_i^l \mid i \in [1,n]\}$.

We use Graph Attention Network v2 (GATv2; Brody et al. 2022) for learning node representations which replaces the static attention mechanism of GAT (Veličković et al., 2018) with a dynamic and more expressive variant. Figure 2.4 shows a GAT layer computing representation for node $h_1^{l-1} \in \mathbb{R}^d$ corresponding to v_1. Let $\mathcal{N}_i = \{v_j \in \mathcal{V} \mid (j,i) \in \mathcal{E}\}$ denote the neighbors of node v_i and α_{ij} the attention score between node h_i and h_j. We calculate attention as a single-layer feed-forward neural network:

$$\alpha_{ij} = \frac{\exp\left(\psi\left(h_i^{l-1}, h_j^{l-1}\right)\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\psi\left(h_i^{l-1}, h_k^{l-1}\right)\right)} \tag{2.19}$$

The scoring function ψ is computed as follows:

$$\psi\left(h_i^{l-1}, h_j^{l-1}\right) = a^T \,\mathrm{LeakyReLU}\left(W \cdot [h_i^{l-1} \oplus h_j^{l-1}]\right) \tag{2.20}$$

where $a \in \mathbb{R}^d$ and $W \in \mathbb{R}^{d \times 2d}$ are learned parameters and d is dimension of each hidden node. $\oplus$ is the concatenation operation. Equation 2.20 adds $\mathrm{LeakyReLU}$ (Maas et al., 2013) non-linearity before softmax computation in Equation 2.19 that keeps the attention dependent on the original h_i^{l-1}. $\mathrm{LeakyReLU}$ is defined through the following equation:

$$\mathrm{LeakyReLU}(x) = \begin{cases} x & \text{if } x > 0, \\ \zeta x & \text{otherwise.} \end{cases} \tag{2.21}$$

Here, ζ is a small coefficient (typically a small positive value like $1e-2$) that provides a non-zero gradient when x is less than or equal to zero, thereby allowing some minimal activation and avoiding the dying neurons issue prevalent in traditional ReLU (Maas

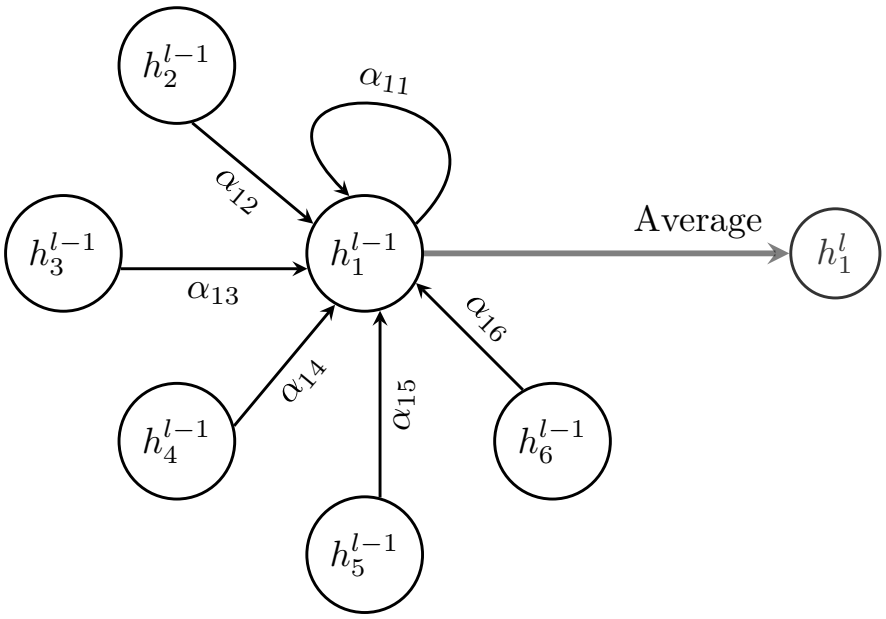

Figure 2.4: A single graph attentional (GAT) layer $l-1$. Nodes are represented as circles represented by dense vectors h_i and directed edges show the flow of information between nodes. α_{ij} is the attention coefficients between nodes h_i and h_j.

et al., 2013) functions. Attention coefficients corresponding to each node i are then used to compute a linear combination of the features corresponding to neighboring nodes as:

$$h_i^l = \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}\bar{W}h_j^{l-1}\right) \tag{2.22}$$

Where, α_{ij} the attention score between node h_i and h_j, σ is the sigmoid non-linearity. $\bar{W}\in\mathbb{R}^{d\times d}$ is the output projection matrix and $h_j^{l-1}\in\mathbb{R}^d$.

2.7 Pretrained Language Models

The best-performing systems in language tasks combine pretraining with task-specific supervised fine-tuning. Various models and architectures have been introduced to achieve this. Initially, word vectors like GLOVE (Pennington et al., 2014) were learned and used as initial word representations in task-specific architectures (see Chapter 3). However, these word representations are static; they capture statistics from the entire corpus, resulting in context-independent representations of individual words. Following ELMO (Peters et al., 2018), BERT (Devlin et al., 2019) introduced the idea of training a bidirectional transformer encoder using a Masked Language Modeling (MLM) objective to learn contextual word representations. Instead of predicting the next word, MLM

involves a fill-in-the-blank or cloze task (Taylor, 1953). This allows the model to learn from context in both directions.

Building on the success of BERT, subsequent research has explored and developed other models, such as BART (Lewis et al., 2020), as well as improved pretraining strategies (Liu et al., 2020b; Wang et al., 2022; Sanh et al., 2022). GPT-2 (Radford et al., 2019), introduced decoder-only (Bengio et al., 2000; Mikolov and Zweig, 2012) transformer models, in contrast to BERT-based architectures that are encoder-only. It demonstrated that language models can perform downstream tasks in a zero-shot setting by feeding input to the model as a prefix, and priming it with keywords such as TL;DR: for summarization, and then autoregressively sampling an output. T5 (Raffel et al., 2020) showed strong transfer learning ability by pretraining an encoder-decoder model. It introduced a unified framework that converts all text-based language tasks into a text-to-text format. These often require fine-tuning the models on downstream tasks (see Chapters 4 and 5).

In parallel to these advances, it has been discovered that scaling either data or model size for pretrained language models improves their performance on various downstream tasks (Kaplan et al., 2020). Based on this observation, many transformer models – especially decoder-only ones – have been scaled, including the 175B-parameter GPT-3 (Brown et al., 2020a) and the 540B-parameter PaLM (Chowdhery et al., 2023). These "large language models" (LLMs) have displayed surprising (*emergent*; Wei et al. (2022a)) abilities in solving a series of complex tasks. For example, GPT-3 can effectively solve few-shot tasks learning, whereas GPT-2 struggles in this regard (Brown et al., 2020a). It is interesting to note that, the choice for decoder-only models stems from their superior zero-shot generalization compared to other variants (Wang et al., 2022). However, due to missing inductive biases in decoder-only models, such as cross-attention (see Section 2.6.3), encoder-decoder models may be well suited for tasks with different input-output modalities. Concurrently, data scaling has been shown to be effective, enabling smaller models to achieve competitive performance (Jiang et al., 2023; Abdin et al., 2024; Team et al., 2024; Dubey et al., 2024).

However, these models struggle with long-range conversational understanding (Maharana et al., 2024). Moreover, it is not obvious how to effectively encode structural information within these models (Fatemi et al., 2024; Huang et al., 2024). In Chapter 6, we will address this issue specifically for the task of conversational question answering and present a method for incorporating structured context representations into pretrained large language models (LLMs), such as Mistral (Jiang et al., 2023).

2.8 Summary

In this chapter we outlined the primary tasks addressed in this thesis. We provide definitions for various types of context and introduced datasets used in our experiments. We also discussed how we evaluate model performance and briefly introduced the core deep learning architectures that we will use as building blocks of our own models. In Chapter 3, we look at conversational question answering based on semantic parsing with relational database as *external* context. We will present a memory-based model for modeling *discourse* context by learning a memory controller in end-to-end manner. We will show that our memory controller effectively manages long-range discourse and is helpful in handling several linguistic phenomenons.

Chapter 3

Discourse Memory for Conversational Semantic Parsing

In the previous chapter, we discussed how understanding conversational language within context demands comprehension of discourse and external context. In this chapter, we investigate *how to process discourse context for understanding extended interactions*. Our hypothesis is that **memory-based representations of discourse are effective in long-range interactive semantic parsing.** We propose to represent discourse information using an external memory. Our memory is *bounded* and does not grow with the length of interaction. In other words, the size of memory remains fixed, independent of the length of the conversation, which is helpful in processing long-range interactions, such as those in the ATIS dataset, which has a maximum of 64 turns. Our memory is interpretable and is managed by a learned controller that is responsible for updating the memory content at each turn. We validate our hypothesis on the context-dependent semantic parsing task using the ATIS (Suhr et al., 2018), SPARC (Yu et al., 2019b), and COSQL (Yu et al., 2019a) datasets. These datasets focus on text-to-SQL generation tasks, which have emerged as a popular application area in recent years (Özcan et al., 2020; Zamani et al., 2022). The datasets contain relational databases, which serve as external contexts. ATIS contains utterances paired with SQL queries pertaining to a US flight booking task; it exemplifies several long-range discourse phenomena, however, it covers a single domain. In contrast, both SPARC and COSQL present cross-domain challenges, requiring the model to generalize to unseen databases at test time, although they involve shorter interaction lengths. We show that our approach can maintain performance over long-range interactions and address several discourse-related phenomena to a large extent.

Q1: What Continental flights go from Chicago to Seattle before 10 am in morning 1993 February twenty sixth

SQL1: (SELECT DISTINCT flight.flight_id FROM flight WHERE (flight.airline_code = 'CO' AND (flight.from_airport IN (SELECT airport_service.airport_code FROM airport_service WHERE airport_service.city_code IN (SELECT city.city_code FROM city WHERE city.city_name = 'CHICAGO')) AND (flight.to_airport IN (SELECT airport_service.airport_code FROM airport_service WHERE airport_service.city_code IN (SELECT city.city_code FROM city WHERE city.city_name = 'SEATTLE')) AND (flight.departure_time < 1000)))))

Q2: Continental flights before noon that have a meal

SQL2: (SELECT DISTINCT flight.flight_id FROM flight WHERE (flight.airline_code = 'CO' AND (flight.from_airport IN (SELECT airport_service.airport_code FROM airport_service WHERE airport_service.city_code IN (SELECT city.city_code FROM city WHERE city.city_name = 'CHICAGO')) AND (flight.to_airport IN (SELECT airport_service.airport_code FROM airport_service WHERE airport_service.city_code IN (SELECT city.city_code FROM city WHERE city.city_name = 'SEATTLE')) AND (flight.departure_time < 1200 AND flight.meal_code IN (SELECT food_service.meal_code FROM food_service WHERE 1 = 1)))))

Q3: Continental flights before 2 pm

SQL3: (SELECT DISTINCT flight.flight_id FROM flight WHERE (flight.airline_code = 'CO' AND (flight.from_airport IN (SELECT airport_service.airport_code FROM airport_service WHERE airport_service.city_code IN (SELECT city.city_code FROM city WHERE city.city_name = 'CHICAGO')) AND (flight.to_airport IN (SELECT airport_service.airport_code FROM airport_service WHERE airport_service.city_code IN (SELECT city.city_code FROM city WHERE city.city_name = 'SEATTLE')) AND (flight.departure_time < 1400 AND flight.meal_code IN (SELECT food_service.meal_code FROM food_service WHERE 1 = 1)))))

Q4: On 1993 February twenty seventh

Q5: All Continental flights leaving Chicago before 8 am on 1993 February twenty seventh

Figure 3.1: Example utterances from a user interaction in the ATIS dataset. Utterance segments referring to the same entity or objects are in same color. SQL queries corresponding to Q3–Q5 follow a pattern similar to Q1 and are not shown for the sake of brevity.

Figure 3.1 displays a sequence of utterances in an interaction. Recall from Chapter 2 that an interaction consists of a sequence of user utterances and the corresponding system responses. In this instance, the user is interested in flight options, and the system's response is an SQL query that, upon execution, provides the answer. Here, the discourse focuses on a specific *topic* serving a specific information need, namely finding out which Continental flights leave from Chicago on a given date and time. Importantly, interpreting each of these utterances, and mapping them to a database query to retrieve an answer needs to be situated in a particular context as the exchange proceeds. The topic further evolves as the discourse transitions from one utterance to the next and constraints (e.g., TIME or PLACE) are added or revised. For example, in Q2 the TIME constraint *before 10am* from Q1 is revised to *before noon*, and in Q3 to *before 2pm*. Aside from such *topic extensions* (Chai and Jin, 2004), the interpretation of Q2 and Q3 depends on Q1, as it is implied that the questions concern Continental flights that go from Chicago to Seattle, not just any Continental flights, however, the phrase *from Chicago to Seattle* is elided from Q2 and Q3. The interpretation of Q4 depends on Q3 which in turn depends on Q1. Interestingly, Q5 introduces information with no dependencies on previous discourse and in this case, relying on information from previous utterances will lead to incorrect SQL queries.

The problem of contextual language processing has been most widely studied within dialogue systems where the primary goal is to incrementally fill pre-defined slot-templates, which can be then used to generate appropriate natural language responses (Gao et al., 2019a). But the rich semantics of SQL queries makes the task of contextual text-to-SQL parsing substantially different. Previous approaches (Suhr et al., 2018; Zhang et al., 2019) tackle this problem by enabling the decoder to copy or modify the *previous* queries under the assumption that they contain all necessary context for generating the current SQL query. The utterance history is encoded in a hierarchical manner and although this is a good enough approximation for most queries (in existing datasets), it is not sufficient to model long-range discourse phenomena (Grosz and Sidner, 1986).

Our own work draws inspiration from Kintsch and van Dijk's (1978) text comprehension model. In their system the process of comprehension involves three levels of operations. First, smaller units of meaning, i.e., propositions, are extracted and organized into a coherent whole (*microstructure*); some of these are stored in a working memory buffer and allow to decide whether new input overlaps with already processed propositions. Second, the gist of the whole is condensed (*macrostructure*). And third,

the previous two operations generate new texts in working with the memory. In other words, the (short and long term) memory of the reader gives meaning to the text read. They propose three macro rules, viz., deletion, generalization, and construction as essential to reduce and organize the detailed information of the microstructure of the text. Furthermore, previous knowledge and experience are central to the interpretation of text enabling the reader to fill information gaps.

Our work borrows several key insights from Kintsch and van Dijk (1978) without being a direct implementation of their model. Specifically, we also break down input utterances into smaller units, namely phrases, and argue that this information can be effectively utilized in maintaining contextual information in an interaction. Furthermore, the notion of a *memory* buffer which can be used to store and process new and old information plays a prominent role in our approach. We propose a **Mem**ory-based **C**ont**E**xt model (which we call MemCE for short) for keeping track of contextual information, and learn a context memory controller that manages the memory. Each interaction (sequence of user utterances) maintains its context using a memory matrix. User utterances are segmented into a sequence of phrases representing either new information to be added into the memory (e.g., *that have a meal* in Figure 3.1) or old information which might conflict with current information in memory and needs to be updated (e.g., *before 10 am* should be replaced with *before noon* in Figure 3.1). Our model can inherently add new content to memory, read existing content by accessing the memory, and update old information.

We evaluate our approach on the ATIS (Suhr et al., 2018; Dahl et al., 1994), SParC (Yu et al., 2019b), and CoSQL (Yu et al., 2019a) datasets. We observe performance improvements when we combine MemCE with existing models underlying the importance of more specialized mechanisms for processing context information. In addition, our model brings interpretability in how the context is processed. We are able to inspect the learned memory controller at test by using hard memory updates ($\max$ in place of $\mathrm{softmax}$, see Section 3.2.2). This allows us to analyze whether important discourse phenomena such as coreference and ellipsis are modeled.

In Section 3.1, we formalize context dependent semantic parsing while Section 3.2 discusses the proposed model in detail. We then evaluate and analyze our approach in Sections 3.4 and 3.5. Next, in Section 3.5.3, we interpret the learned memory controller by plotting the contents of our memory module.

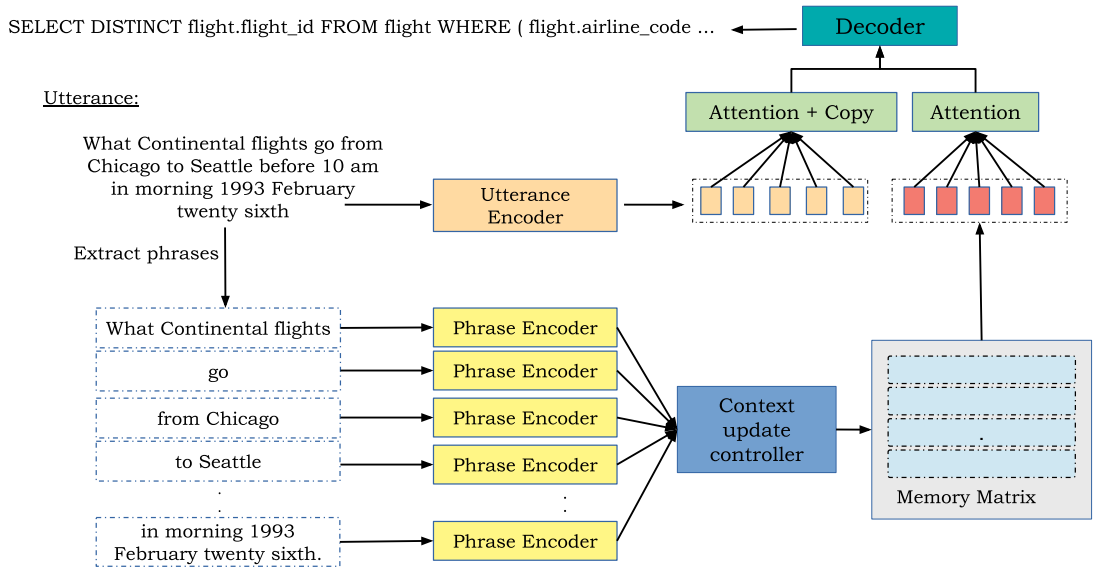

Figure 3.2: Overview of the model architecture. Utterances are broken down into segments. Each segment is encoded with the same encoder (same weights) and is processed independently. Utterance Encoder encodes the current utterance. The context update controller learns to manipulate the memory such that correct discourse information is retained. The decoder attends to both the memory and the current utterance representation to generate the semantic parse.

3.1 Problem Formulation

We revisit from Chapter 2 (Section 2.2) the problem formulation for context-dependent semantic parsing within conversational settings. We focus on interactions involving relational databases as external context denoted as C. We denote an interaction as $I = [X_t, Z_t, A_t]_{t=1}^{T}$. Where X_t represents a user's utterance, Z_t is the semantic parse, and each A_t is the corresponding answer at turn t. Each user utterance X_t and the corresponding system answer A_t are composed of a sequence of natural language tokens, expressed as $\langle x_1, x_2, \ldots, x_{|X_t|} \rangle$ and $\langle a_1, a_2, \ldots, a_{|A_t|} \rangle$ respectively. In contrast, Z_t includes a sequence of tokens $\langle z_1, z_2, \ldots, z_{|Z_t|} \rangle$. These tokens differ from those in X_t and A_t in that every token z_i may either be a natural language word or a SQL keyword. We learn a semantic parsing model $\mathcal{M}$ with parameters Θ that transforms a user's utterance X_t into a formal SQL representation Z_t.

$$Z_t = \mathcal{M}\left(I[:t-1], X_t, \mathsf{C}; \Theta\right) \tag{3.1}$$

Here $I[:t-1]$ denote past interactions and answer A_t is obtained upon executing Z_t on the relational database C.

$$A_t = \text{Execute}\,(Z_t, \text{C}) \tag{3.2}$$

The Execute operator denotes the query engine that executes the parsed input Z_t to produce the output A_t. In the cross-domain context assumed by both SPARC and COSQL, each interaction I potentially involves a different database C. Additionally, at test time, the model must generalize to databases unseen during training. In the next section we describe the proposed model $\mathcal{M}$.

3.2 Model

Our model is based on the encoder-decoder architecture (Cho et al., 2015) with the addition of a memory component (Sukhbaatar et al., 2015; Santoro et al., 2016) for incorporating context. As shown in Figure 3.2, our model consists of four components, (1) a memory matrix retains discourse information, (2) a memory controller, learns to access and manipulate the memory such that correct discourse information is retained, (3) utterance and phrase encoders, and (4) a decoder which interacts with the memory and utterance encoder using an attention mechanism to generate SQL output.

3.2.1 Input Encoder

Each input utterance $X_t = (x_{t,1} \ldots x_{t,|X_t|})$ is encoded using a bi-directional LSTM (Hochreiter and Schmidhuber, 1997),

$$h^U_{t,j} = \text{biLSTM}^U(e_{t,j}; h^U_{t,j-1}) \tag{3.3}$$

where, $e_{t,j} = \phi(x_{t,j})$ is a learned embedding corresponding to input token $x_{t,j}$ and $h^U_{t,j}$ is the concatenation of the forward and backward LSTM (described in Chapter 2 Section 2.6.1) hidden representations at step j. As mentioned earlier, X_t is also segmented into a sequence of phrases denoted as $X_t = (p^1_t \ldots p^K_t)$, where K is the number of phrases for utterance X_t. We provide details on how utterances are segmented into phrases in Section 3.3. For now, suffice it to say that they are obtained from the output of a chunker with some minimal postprocessing (e.g., to merge postmodifiers with NPs or VPs). Each phrase consists of tokens $p^k_t = (x_{t,[s_k:s_k+|p^k_t|]})$, such that $k \in [1,K]$ and $s_k = \sum_{z=1}^{k-1} |p^z_t|$. Each phrase p^k_t is separately encoded using a bi-directional LSTM,

$$h^P_{t,k,j} = \text{biLSTM}^P(e_{t,j}; h^P_{t,k,j-1}) \tag{3.4}$$

such that $j \in [s_k : s_k + |p_t^k|]$. As shown in Figure 3.2, every phrase p_t^k in utterance t is separately encoded using biLSTM^P to obtain a phrase representation $h_{t,k}^P$ by concatenating the final forward and backward hidden representations.

3.2.2 Context Memory

Our context memory is a matrix $M_t \in \mathbb{R}^{L \times d}$ with L memory slots, each of dimension d, where M_t is the state of the memory matrix at the t^{th} interaction turn. The goal of context memory is to maintain relevant information required to parse the input utterance at each turn. As shown in Figure 3.2, this is achieved by learning a *context update controller* which is responsible for updating the memory at each turn.

For each phrase p_t^k belonging to a sequence of phrases within utterance X_t, the controller decides whether it contains old information which conflicts with information present in the memory or new information which has to be added to the current context. When novel information is introduced, the controller should add it to an empty or least-used memory slot, otherwise the conflicting memory slot should be updated with the latest information. Let τ denote the memory update time step such that $\tau \in [1, n]$, where n is the total number of phrases in interaction I. We simplify notation, using h_τ^P instead of $h_{t,k}^P$, to represent the hidden representation of a phrase at time τ. The memory state at update step τ is denoted by M_t^τ.

Detecting Conflicts Given phrase representation h_τ^P (see Equation (3.4)), we use a similarity module to detect conflicts between h_τ^P and every memory slot in $M_t^\tau(m)$ where $m \in [1, L]$; $M_t^\tau(m)$ is the m^{th} row representing a memory slot in the memory matrix. Intuitively, low similarity represents new information. Our similarity module is based on a Siamese network architecture (Bromley et al., 1994) that takes phrase hidden representation h_τ^P and memory slot $M_t(m)$ and computes a low-dimensional representation using the same neural network weights. The resulting low-dimensional representations are then compared using the cosine distance metric:

$$\hat{w}_c^{\tau,m} = \frac{\text{sia}(h_\tau^P) \cdot \text{sia}(M_t^{\tau-1}(m))}{\max(\| \text{sia}(h_\tau^P) \|_2 \cdot \| \text{sia}(M_t^{\tau-1}(m)) \|_2, \varepsilon)} \tag{3.5}$$

where ε is a small value for numerical stability and sia is a multi-layer feed-forward network with a $\tanh$ activation function. For hidden representation h, sia is computed as:

$$\hat{h} = W(\tanh(W^l h + b^l) + b) \tag{3.6}$$

where l represents the layer number and W^l, b^l, W, and b are learnable parameters. We use $\hat{w}_c^{\tau,m}$ to obtain a similarity distribution w_s^τ for updating step τ over memory slots. w_s^τ represents the probability of dissimilarity (or conflict) which is calculated by computing $\mathrm{softmax}$ over cosine similarities with every memory slot $m \in [1..L]$:

$$w_s^\tau = \mathrm{softmax}([\hat{w}_c^{\tau,1}; \hat{w}_c^{\tau,2} \ldots; \hat{w}_c^{\tau,L}]) \tag{3.7}$$

We compute $\mathrm{softmax}$ over cosine values so that the linear combination of w_s^τ with least used weights w_{lu}^τ (described below in the memory update paragraph) still represents the probability of update across each memory slot.

Adding New Information To add new information to the memory, i.e., when there is no conflict with any locations, we need to ascertain which memory locations are either empty or rarely used. When the memory is full, i.e., all memory slots are used during previous updates, we update the slot which was least used. This is accomplished by maintaining memory usage weights $w_u^\tau \in \mathbb{R}^L$ at each update τ; w_u^τ is initialized with zeros at $\tau = 0$ and is updated by combining previous memory usage weights $w_u^{\tau-1}$ with current write weight w_w^τ using a decay parameter λ:

$$w_u^\tau = w_w^\tau + \lambda w_u^{\tau-1} \tag{3.8}$$

where write weights w_w^τ are used to compute the write location and are described in the memory update paragraph below. The least used weight vector w_{lu}^τ, at update step τ is then calculated as:

$$w_{lu}^\tau = \mathrm{softmin}(w_u^{\tau-1}) \tag{3.9}$$

where for vector x we calculate $\mathrm{softmin}(x) = \exp(-x)/\sum_j \exp(-x_j)$. Hard updates, i.e., using $\mathrm{smallest}$ instead of $\mathrm{softmin}$ are also possible. However, we found $\mathrm{softmin}$ to be more stable during learning.

Memory Update We wish to compute write location w_w^τ given least used weight vector w_{lu}^τ and conflict probability distribution w_s^τ. Notice that w_s^τ and w_{lu}^τ are essentially two probability distributions each representing a candidate write location in memory. We learn a convex combination parameter μ which depends on w_s^τ,

$$\mu = \sigma(W_\sigma w_s^\tau + b_\sigma) \tag{3.10}$$

$$w_w^\tau = \mathrm{softmax}((\mu w_s^\tau + (1-\mu) w_{lu}^\tau)/\Delta) \tag{3.11}$$

where temperature hyperparameter Δ is used to concentrate the update to a single location.

Finally, the memory is updated with current phrase representation h_τ^P as,

$$M_i^\tau(m) = M_i^{\tau-1}(m) + w_w^\tau(m) h_\tau^P \quad \forall m \in [1, L] \tag{3.12}$$

3.2.3 Decoder

The output query is generated with an LSTM decoder. As shown in Figure 3.2, the decoder depends on the memory and utterance representations computed using Equations (3.12) and (3.3), respectively. The decoder state at the decoding step s is computed as:

$$h_s^D = \text{LSTM}([\phi^o(z_{t,s-1}); c_{s-1}^M; c_{s-1}^U]; h_{s-1}^D) \tag{3.13}$$

where ϕ^o is a learned embedding function for output tokens, $z_{t,s-1}$ is the output token at step $s-1$, c_{s-1}^U is an utterance context vector, c_{s-1}^M is a memory context vector, and h_{s-1}^D is the previous decoder hidden state $s-1$. c_s^U is calculated as the weighted sum of all hidden states,

$$v_s(j) = h_{t,j}^U W^A h_s^D \tag{3.14}$$

$$\alpha_s^U = \text{softmax}(v_s) \tag{3.15}$$

$$c_s^U = \sum_j h_{t,j}^U \alpha_s^U(j) \tag{3.16}$$

Where α_s^U is the utterance state attention score. Memory state attention score α_s^M and memory context vector c_s^M are computed in a similar manner using memory slots as hidden states. The probability of output query tokens is computed as:

$$P(\hat{z}_{t,s} | X_t, I[:t-1]) \propto \exp(\tanh([h_s^D; c_s^U; c_s^M] W^{\hat{o}}) W^o + b^o) \tag{3.17}$$

We further modify the decoder in order to deal with the large number of database values (e.g., city names) common in text-to-SQL semantic parsing tasks. As in Suhr et al. (2018), we add anonymized token attention scores in the output vocabulary distribution which enables copying anonymized tokens mentioned in input utterances. The final probability distribution over output vocabulary tokens and anonymized tokens is:

$$P(z_{t,s}) = \text{softmax}(P(\hat{z}_{t,s}) \oplus P(\hat{a}_{t,s})) \tag{3.18}$$

where $\oplus$ represents concatenation and $P(\hat{a}_{t,s})$ are anonymized token attention scores in the attention distribution α_s^U.

3.2.4 Training

Our model is trained in an end-to-end fashion using a cross-entropy loss. Given a training set of N interactions $\{I^{(l)}\}_{l=1}^{N}$, such that each interaction $I^{(l)}$ consists of utterances $X_t^{(l)} = (x_{t,1}^{(l)} \ldots x_{t,|X_i^{(l)}|}^{(l)})$ paired with output queries $Y_t^{(l)} = (y_{t,1}^{(l)} \ldots y_{t,|Y_t^{(l)}|}^{(l)})$, we minimize token cross-entropy loss as:

$$\mathcal{L}(\hat{z}_{t,k}^{(l)}) = -logP(\hat{z}_{t,k}^{(l)} | x_t^{(l)}, z_{t,k}^{(l)}, I[:t-1]) \tag{3.19}$$

where, $\hat{z}_{t,k}^{(l)}$ denotes the predicted output token and k is the gold output token index. The total loss is the average of the utterance level losses used for back-propagation.

3.3 Experimental Setup

We evaluated MemCE, our memory-based context model, on various settings by integrating it with multiple open-source models. We achieve this by replacing the discourse component of related models with MemCE subject to minor or no additional changes. All base models in our experiments use a turn-level hierarchical encoder to capture previous language context. For primary evaluation, we use the ATIS (Hemphill et al., 1990; Dahl et al., 1994) dataset but also present results on SParC (Yu et al., 2019b) and CoSQL (Yu et al., 2019a). We refer the reader to Chapter 2 (Section 2.4) for examples and details about these datasets.

Utterance Segmentation We segment each input utterance into a sequence of phrases with a pretrained chunker and then apply a simple rule-based merging procedure to create bigger chunks as an approximation to propositions (Kintsch and van Dijk, 1978). Figure 3.3 illustrates the process. We used the Flair chunker (Akbik et al., 2018) trained on Conll-2000 (Tjong Kim Sang and Buchholz, 2000) to identify NP and VP phrases without postmodifiers. Small chunks (e.g., *from, before* in the figure) were subsequently merged into segments using the following rules and NLTK's (Bird et al., 2009) tag-based regex merge:

R1: $left = \langle VP.* \rangle, right = \langle VP.* \rangle$

R2: $left = \langle PP.* \rangle | \langle NP.* \rangle, right = \langle NP \rangle +$

R3: $left = \langle NP.* \rangle, right = \langle VB.* \rangle$

R4: $left = \langle AD.* \rangle, right = \langle NP.* \rangle$

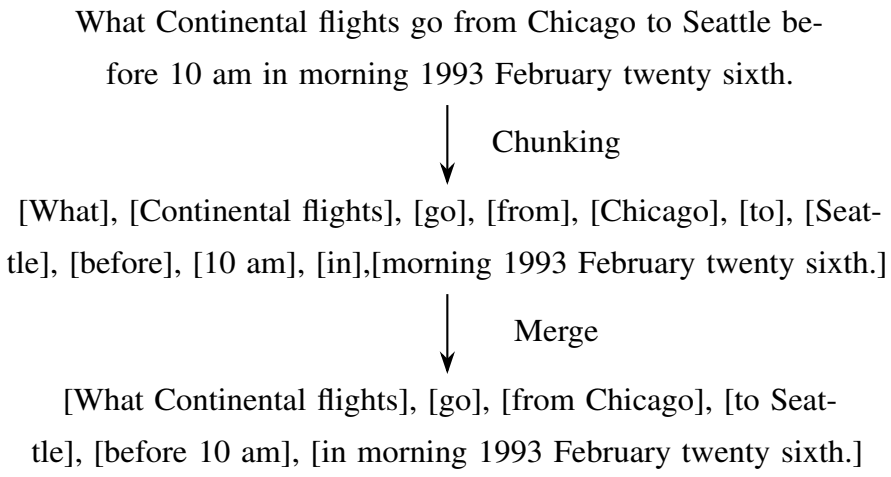

Figure 3.3: Example of sentence segmentation using chunking and rule-based merging. First step uses a pretrained chunker which are then merged using simple rules.

The rules above are applied in order. For each rule we find any chunk whose end matches the left pattern followed by a chunk whose beginning matches the right pattern. Chunks that satisfy this criterion are merged. In an effort to keep the rule set as small as possible, we allowed verbs like *go* to be processed as independent units.

We segment utterances and anonymize entities independently and then match entities within segments deterministically. This step is necessary to robustly perform anonymization as in some rare cases, the chunking process will separate entities in two different phrases (e.g., *in Long Beach California that* is chunked as *in Long Beach* and *California that*). This is easily handled by a simple token number matching procedure between the anonymized utterance and corresponding phrases.

Model Configuration Our model is implemented in PyTorch (Paszke et al., 2019a). For all experiments, we used the ADAM optimizer (Kingma and Ba, 2015a) to minimize the loss function and the initial learning rate was set to 0.001. During training, we used the ReduceLROnPlateau learning rate scheduling strategy on the validation loss, with a decay rate of 0.8. We also applied dropout with 0.5 probability. Dimensions for the word embeddings were set to 300. Following previous work (Zhang et al., 2019) we use pretrained GloVe (Pennington et al., 2014) embeddings for our main experiments on the SparC and CoSQL datasets. For ATIS, word embeddings were not pretrained (Suhr et al., 2018; Zhang et al., 2019). Memory length was chosen as a hyperparameter from the range [15, 25] and the temperature parameter was chosen

from $\{0.01, 0.1\}$. Best memory length values for ATIS, SparC, and CoSQL were 25, 16, and 20, respectively. This correlates with the average interaction length of the individual datasets (see Table 2.4). The RNN decoder is a two-layer LSTM and the encoder is a single layer LSTM. The Siamese network in the module which detects conflicting slots uses two hidden layers. We used development set of the individual datasets for hyperparameter selection.

3.4 Results

In this section, we assess the effectiveness of the MemCE encoder at handling contextual information. We present our results, evaluation methodology, and comparisons against the state of the art.

3.4.1 Evaluation on ATIS

We primarily focus on ATIS because it contains relatively long interactions (average length is 7) compared to other datasets (e.g, the average length in SParC is 3). Longer interactions present multiple challenges that require non-trivial processing of context, some of which are discussed in Section 3.5. We use the ATIS dataset split created by Suhr et al. (2018). It contains 27 tables and 162K entries with 1,148/380/130 train/dev/test interactions. The semantic representations are in SQL.

Following Suhr et al. (2018), we measure *query accuracy*, *strict denotation accuracy* (see Chapter 2 Section 2.5), and *relaxed denotation accuracy*. Query accuracy is the percentage of predicted queries that match the reference query. Strict denotation accuracy is the percentage of predicted queries that when executed produce the same results as the reference query. Relaxed accuracy also gives credit to a prediction query that fails to execute if the reference table is empty. In cases where the utterance is ambiguous and there are multiple gold queries, the query or table is considered correct if they match any of the gold labels. We evaluate on both development and test set, and select the best model during training via a separate validation set consisting of 5% of the training data.

Table 3.1 presents a summary of our results. We compare our approach against a simple Seq2Seq model which is a baseline encoder-decoder without any access to contextual information. Seq2Seq+Concat is a strong baseline which consists of an encoder-decoder model with attention on the current and the *previous three concatenated*

Model	Enc-Dec	Dev Set			Test Set		
			Denotation			Denotation	
		Query	Relaxed	Strict	Query	Relaxed	Strict
Seq2Seq	LSTM-LSTM	28.7	48.8	43.2	35.7	56.4	53.8
Seq2Seq+Concat	LSTM-LSTM	35.1	59.4	56.7	42.2	66.6	65.8
Suhr et al. (2018)	HE-LSTM	36.0	59.5	58.3	—	—	—
Suhr et al. (2018)	HE-SnipCopy	37.5	63.0	62.5	43.6	69.3	69.2
Zhang et al. (2019)	HE-EditBased	36.2	60.5	60.0	43.9	68.5	68.1
Lin et al. (2019)	LSTM-Grammar	39.1	—	**65.8**	44.1	—	**73.7**
MemCE	Mem-LSTM	**40.2**	63.6	61.2	**47.0**	70.1	68.9
MemCE	Mem-SnipCopy	39.1	**65.5**	65.2	45.3	**70.2**	69.8

Table 3.1: Model accuracy on the ATIS dataset. HE is a hierarchical interaction encoder, while Mem is the proposed memory-based encoder. LSTM are vanilla encoder/decoder models, while SnipCopy copies SQL segments from the previous query and EditBased adopts a query editing mechanism. Memory-based experiment accuracy is averaged over five runs.

utterances. We also compare against the models of Suhr et al. (2018) and Zhang et al. (2019). The former employs a turn-level encoder on top of an utterance-level encoder in a *hierarchical* fashion together with a decoder which learns to copy complete SQL segments from the previous query (SQL segments between consecutive queries are aligned during training using a rule-based procedure). The turn-level encoder maintains a dense vector to carry the discourse state from the previous utterances. The latter enhances the turn-level encoder by employing an attention mechanism across different turns. They propose turn attention mechanism that captures correlations between the current utterance and previous ones by computing dot-product attention. It additionally introduces a *query editing* mechanism which decides at each decoding step whether to copy from the previous query or insert a new token.

Column Enc-Dec in Table 3.1 describes the various models in terms of the type of encoder/decoder used. LSTM is a vanilla encoder or decoder, HE is a turn-level hierarchical encoder (Suhr et al., 2018), and Mem is the proposed memory-based encoder. SnipCopy and EditBased respectively refer to Suhr et al.'s 2018 and Zhang et al.'s 2019 decoders. We present two instantiations of our MemCE model with a simple LSTM decoder (Mem-LSTM) and SnipCopy (Mem-SnipCopy). Table 3.1 also reports the results from Lin et al. (2019) [1] who apply a grammar-based decoder to this task; they also incorporate the interaction history by concatenating the current utterance with the previous three utterances which are encoded with a bi-directional LSTM. All models in Table 3.1 use entity anonymization, Lin et al. (2019) additionally use identifier linking, i.e., string matching heuristic rules to link words or phrases in the input utterance to identifiers in the database (e.g., `city_name_string -> ''BOSTON''`).

As shown in Table 3.1, MemCE is able to outperform comparison systems. We observe a boost in denotation accuracy when using the SnipCopy decoder instead of an LSTM-based one, however, exact match does not improve. This is possibly because SnipCopy makes it easier to generate long SQL queries by copying segments, but at the same time it suffers from spurious generation and error propagation.

Table 3.2 presents various ablation studies which evaluate the contribution of individual model components. We use Mem-SnipCopy as our base model and report performance on the ATIS development set following the configuration described in Section 3.3. We first remove the proposed memory controller described in Section 3.2.2 and

[1] We relied on the results published in the arXiv paper, as our own experiments did not replicate them.

		Denotation	
	Query	Relaxed	Strict
MemCE+SnipCopy	39.1	65.5	65.2
Without memory controller	34.3	58.7	58.1
Phrases are utterance tokens	37.2	61.9	61.7
Phrases are full utterances	36.8	64.2	63.9

Table 3.2: Ablation results with SnipCopy decoder on the ATIS development set.

	Train	Dev	Test
#Interactions	2869	290	290
#Utterances	8535	851	821

Table 3.3: Statistics for SParC-DI domain-independent split which has 157 domains in total.

simplify Equation (3.11) using key-value based attention to calculate w_w^τ as,

$$\alpha_j = M_t^{\tau-1}(j) W^P h_\tau^P \tag{3.20}$$

$$w_w^\tau = \text{softmax}(\alpha) \tag{3.21}$$

We observe a decrease in performance (see second row in Table 3.2) indicating that the proposed memory controller is helpful in maintaining interaction context.

We performed two ablation experiments to evaluate the usefulness of utterance segmentation. First, instead of the phrases extracted from our segmentation procedure, we employ a variant of our model which operates over individual tokens (see row "phrases are utterance tokens" in Table 3.2). As can be seen, this strategy is not optimal as results decrease across metrics. We believe operating directly on tokens can lead to ambiguity during update. For example, when processing the phrase *to Boston* given previous utterance *What Continental flights go from Chicago to Seattle*, it is not obvious whether *Boston* should update *Chicago* or *Seattle*. Second, we do not use any segmentation at all, not even at the token level. Instead, we treat the entire utterance as a single phrase (see row "phrases are full utterances" in Table 3.2). If memory's only function is to simply store utterance encodings, then this model becomes comparable to a hierarchical encoder with attention. Again, we observe that performance decreases which indicates that our system benefits from utterance segmentation. Overall, the ablation studies in Table 3.2 show that segmentation and its granularity matters. Our heuristic procedure

works well for the task at hand, although a learning-based method would be more flexible and potentially lead to further improvements.

3.4.2 Evaluation on SParC and CoSQL

In this section we describe our results on SParC and CoSQL. Both datasets assume a cross-domain semantic parsing task in context with SQL as the meaning representation. In addition, for ambiguous utterances, (which cannot be uniquely mapped to SQL given past context) CoSQL also includes clarification questions (and answers). We do not tackle these explicitly but consider them part of the utterance preceding them (e.g., *please list the singers — did you mean list their names? — yes*). Since our primary objective is to study and measure context-dependent language understanding, we created a split of SParC which is denoted as SParC-DI[2] where domains are all seen in training, development, *and* test set. In this way we ensure that no model has the added advantage of being able to handle cross-domain instances while lacking context-dependent language understanding. Table 3.3 shows the statistics of our SParC-DI split, following a ratio of 80/10/10 percent for the training/development/test set.

We evaluate model output using exact set match accuracy (Yu et al., 2019b).[3] We report two metrics: *question accuracy* which is the accuracy considering all utterances independently, and *interaction accuracy* which is the correct interaction accuracy averaged across interactions. An interaction is marked as correct if all utterances in that interaction are correct. Since utterances in an interaction can be semantically complete (i.e., independent of context), we prefer interaction accuracy.

Table 3.4 summarizes our results. CDS2S is the context-dependent cross-domain parsing model of Zhang et al. (2019). It is is adapted from Suhr et al. (2018) to include a schema encoder which is necessary for SparC and CoSQL. It also uses a turn-level hierarchical encoder to represent the interaction history. We also report model variants where the CDS2S encoder is combined with an LSTM-based encoder, SnipCopy (Suhr et al., 2018) and a grammar-based decoder Liu et al. (2020a). The latter decodes SQL queries as a sequence of grammar rules, rather than tokens. We compare the above systems with three variants of our MemCE model which differ in their use of an LSTM decoder, SnipCopy, and the Grammar-based decoder of Liu et al. (2020a).

Across models and datasets we observe that MemCE improves performance which

[2]We only considered training and development instances as the test set is not publicly available.

[3]Predicted queries are decomposed into different SQL clauses and scores are computed for each clause separately.

Model	Enc-Dec	CoSQL(D)		CoSQL(T)		SparC(D)		SparC(T)		SparC-DI(T)	
		Q	I	Q	I	Q	I	Q	I	Q	I
CDS2S	HE-LSTM	13.8	2.1	13.9	2.6	21.9	8.1	23.2	7.5	39.5	20.1
CDS2S	HE-SnipCopy	12.3	2.1	—	—	21.7	9.5	20.3	8.1	38.7	24
Liu et al. (2020a)	HE-Grammar	**33.5**	9.6	—	—	41.8	20.6	—	—	**57.1**	35.3
MemCE+CDS2S	Mem-LSTM	13.4	3.4	—	—	21.2	8.8	—	—	41.3	22.9
MemCE+CDS2S	Mem-SnipCopy	13.1	2.7	—	—	21.4	10.9	—	—	41.5	26.7
MemCE+Liu et al. (2020a)	Mem-Grammar	32.8	**10.6**	**28.4**	**6.2**	**42.4**	**21.1**	**40.3**	**16.7**	55.7	**36.3**

Table 3.4: Query (Q) and Interaction (I) accuracy for SParC and CoSQL. We report results on the development (D) and test (T) sets. Sparc-DI is our domain-independent split of SparC. CDS2S is the context-dependent cross-domain parsing model of Zhang et al. (2019). HE is a hierarchical encoder and Mem is the proposed memory-based context encoder. LSTM is a vanilla decoder, SnipCopy copies SQL segments from the previous query, and Grammar refers to a decoder which outputs a sequence of grammar rules rather than tokens. Table cells are filled with — whenever results are not available.

	MemCE		Suhr et al. (2018)	
	Denotation	Query	Denotation	Query
Focus Shift	80.4	50.0	76.7	44.6
Referring Exp	80.0	40.0	70.0	20.0
Ellipsis	69.4	33.3	66.6	25.0
Independent	81.4	61.1	81.3	62.7

Table 3.5: Model accuracy on specific phenomena (20 interactions, ATIS dev set).

suggests that it better captures contextual information as an independent language modeling component. We observe that benefits from our memory-based encoder persist across domains and data splits even when sophisticated strategies like grammar-based decoding are adopted.

3.5 Analysis

In this section, we analyze our model's ability to handle important discourse phenomena such as focus shift, referring expressions, and ellipsis. We also showcase its interpretability by examining the behavior of the (learned) memory controller.

3.5.1 Focus Shift

Our linguistic analysis took place on 20 interactions[4] randomly sampled from the ATIS development set (134 utterances in total). Table 3.5 shows overall performance statistics for MemCE (Mem-LSTM) and Suhr et al. (2018) (HE-SnipCopy) on our sample. We annotated the focus of attention in each utterance (underlined in the example below) which we operationalized as the most salient entity (e.g., city) within the utterance (Grosz et al., 1995). Focus shift occurs when the attention transitions from one entity to another. In the interaction below the focus shifts from *flights* in Q2 to *cities* in Q3.

Q1: What flights are provided by American airlines
Q2: What flights are provided by Delta airlines
Q3: Which cities are serviced by both American and Delta airlines

[4]Interactions with less than two utterances were discarded.

Handling focus shift has been problematic in the context of semantic parsing (Suhr et al., 2018). In our sample, 41.8% of utterances displayed focus shift. Our model was able to correctly parse all utterances in the interaction above and is more apt at handling focus shifts compared to related systems (Suhr et al., 2018). Table 3.5 reports denotation and query accuracy on our analysis sample.

3.5.2 Referring Expressions and Ellipsis

Ellipsis refers to the omission of information from an utterance that can be recovered from the context. Consider the following interaction.

Q1: Please give me all flights from Long Beach to Memphis
Q2: What about 1993 June thirtieth
Q3: How about any day
Q4: Where do they stop

Here, Q2 and Q3 exemplify nominal ellipsis, the NP *all flights from Long Beach to Memphis* is elided and ideally should be recovered from the discourse, in order to generate correct SQL queries. Q4 is an example of coreference, *they* refers to the answer of Q3. However, it can also be recovered by considering all previous utterances (i.e., Where do they [flights from Long Beach to Memphis; any day] stop). Since our model explicitly stores information in context, it is able to parse utterances like Q2 and Q4 correctly.

In our ATIS sample, 26.8% of the utterances exhibited ellipsis and 7.5% contained referring expressions. Results in Table 3.5 show that MemCE is able to better handle both such cases. However, we also observe that ellipsis is challenging for both the models.

3.5.3 Memory Interpretation

In this section we inspect the memory controller with the aim of understanding what kind of patterns it learns and where it fails. In Figure 3.4, we visualize the content of memory for an interaction (top row) from the ATIS development set consisting of seven utterances.[5] Each column in Figure 3.4 shows the content of memory after processing

[5]Q4 was repeated in the dataset. We do the same to maintain consistency and to observe the effect of repetition.

Memory slot num	*Q1*: Continental airlines on 1993 February twenty Seventh from Chicago to Seattle	*Q2*: Show 1993 February twenty eighth flights from Seattle to Chicago	*Q3*: Only flights after 1700 hours	*Q4*: Only flights after 1700 hours	*Q5*: Only flights after 1500 hours	*Q6*: Show all 1993 February twenty eighth flights on Continental	*Q7*: Only those on Continental airlines
0	Continental airlines	Continental airlines	Only flights	Only flights	Only flights	Only flights	Only those
1	on 1993 February twenty	Show 1993 February twenty eighth flights	Show 1993 February twenty eighth flights	Show 1993 February twenty eighth flights	Show 1993 February twenty eighth flights	Show all 1993 February twenty eighth flights	Show all 1993 February twenty eighth flights
2	from Chicago	from Seattle	from Seattle	from Seattle	from Seattle	from Seattle	from Seattle
3	to Seattle	to Chicago	to Chicago	to Chicago	to Chicago	to Chicago	to Chicago
4	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
...	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
14	ϕ	ϕ	after 1700 hours	after 1700 hours	after 1500 hours	after 1500 hours	after 1500 hours
15	ϕ	ϕ	ϕ	ϕ	ϕ	on Continental	on Continental
	✓	✗	✓	✓	✓	✗	✗

Figure 3.4: Visualization of memory matrix. Rows represent memory content and columns represents the utterance time step. The top row shows the utterances being processed. Each row is marked with a memory slot number which represents the content of memory in that slot. Empty slots are marked with ϕ. The bottom row shows whether the utterance was parsed correctly(✓) or not(✗). : Stale content in memory w.r.t the current utterance. : Incorrect substitution.

the corresponding utterance in the interaction. The bottom row indicates whether the final output was correct (✓) or not (✗). For the purpose of clear visualization we took the $\max$ instead of $\mathrm{softmax}$ in Equation (3.10) to obtain the memory state at any time step.

Q2 presents an interesting case for our model, it is not obvious whether *Continental airlines* from Q1 should be carried forward while processing Q2. The latter is genuinely ambiguous, it could be referring to Continental airlines flights or to flights by any carrier leaving from Seattle to Chicago. If we assume the second interpretation, then Q2 is more or less semantically complete and independent of Q1. 44% of utterances in our ATIS sample are semantically complete. Although we do not explicitly handle such utterances, our model is able to parse many of them correctly because they usually repeat the

information mentioned in the previous discourse as a single query (see Table 3.5). Q2 also shows that the memory controller is able to learn the similarity between long phrases: *on 1993 February twenty Seventh* ⇔ *Show 1993 February twenty eighth flights*. It also demonstrates a degree of semantic understanding, i.e., it replaces *from Chicago* with *from Seattle* in order to process utterance Q2, rather than simply relying on entity matching.

Figure 3.4 further shows the kind of mistakes the controller makes which are mostly due to stale content in memory. In utterance Q6 the memory carries over the constraint *after 1500 hours* from the previous utterance which is not valid since Q6 explicitly states *Show all . . . flights on Continental*. At the same time constraints *from Seattle* and *to Chicago* should carry forward. Knowing which content to keep or discard makes the task challenging.

Another cause of errors relates to reinstating previously nullified constraints. In the interaction below, Q3 reinstates *from Seattle to Chicago*, the focus shifts from flights in Q1 to ground transportation in Q2 and then again to flights in Q3.

Q1: Show flights from Seattle to Chicago
Q2: What ground transportation is available in Chicago
Q3: Show flights after 1500 hours

Handling these issues altogether necessitates a non-trivial way of managing context. Given that our model is trained in an end-to-end fashion, it is encouraging to observe a one-to-one correspondence between memory and the final output which supports our hypothesis that explicitly modeling language context is helpful.

3.6 Related Work

Sequence-to-sequence neural networks (Bahdanau et al., 2015b) have emerged as a general modeling framework for semantic parsing, achieving impressive results across different domains and semantic formalisms (Dong and Lapata 2016; Jia and Liang 2016; Iyer et al. 2017; Wang et al. 2020; Zhong et al. 2018; Yu et al. 2018, *inter alia*). The majority of existing work has focused on mapping natural language utterances into machine-readable meaning representations *in isolation* without utilizing context information. While this is useful for environments consisting of one-shot interactions of users with a system (e.g., running QA queries on a database), many settings require

extended interactions between a user and an automated assistant (e.g., booking a flight). This makes the one-shot parsing model inadequate for many scenarios.

In this work we are concerned with the lesser studied problem of *contextualized* semantic parsing where previous utterances are taken into account in the interpretation of the current utterance. Earlier work (Miller et al., 1996; Zettlemoyer and Collins, 2009; Srivastava et al., 2017) has focused on symbolic features for representing context, e.g., by explicitly modeling discourse referents, or the flow of discourse. Recent neural methods extend the sequence-to-sequence architecture to incorporate contextual information either by modifying the encoder or the decoder. Context-aware encoders resort to concatenating the current utterance with the utterances preceding it (Suhr et al., 2018; Zhang et al., 2019) or focus on the history of the utterances most relevant to the current decoder state (Liu et al., 2020a). The decoders take context representations as additional input and often copy segments from the previous query (Suhr et al., 2018; Zhang et al., 2019). Hybrid approaches (Iyyer et al., 2017b; Guo et al., 2019; Liu et al., 2020a; Lin et al., 2019) employ neural networks for representation learning but use a grammar for decoding (e.g., a sequence of actions or an intermediate representation).

More recent work has explored in-context learning using Large Language Models (LLMs) for text-to-SQL. Rajkumar et al. (2022) evaluated few-shot learning capabilities of GPT-3 (Brown et al., 2020b) and Codex (Chen et al., 2021) on the Spider (Yu et al., 2018) dataset. DIN-SQL (Pourreza and Rafiei, 2023) employs a task-decomposition approach to enhance reasoning. Their findings indicate that breaking down the generation problem into sub-problems and feeding the solutions into LLMs can significantly improve performance. Concurrent works tackles schema linking and handling multi-turn interactions required by SPARC and COSQL by prompting LLMs using chain-of-thought (CoT; Wei et al. (2022b)). However, creating CoT examples requires manually writing reasoning traces. ACT-SQL (Zhang et al., 2023) automatically generates CoT examples for schema linking. They handle multi-turn dependencies by rewriting queries to make them independent of prior context. However, this approach suffers from error propagation, where errors in rewriting process leads to incorrect SQL generation. CoE-SQL (Zhang et al., 2024a) uses an edit-based method and propose to modify previously generated queries instead of generating it from scratch. However, unlike our approach, these solutions do not explicitly address long-range interactions. Maharana et al. (2024) demonstrate that models such as GPT-3 (Brown et al., 2020a) still struggle to fully comprehend lengthy conversations and the long-range dependencies.

A tremendous amount of work has taken place in the context of discourse modeling fo-

cusing on extended texts (Mann and Thompson, 1988; Hobbs, 1985) and dialogue (Grosz and Sidner, 1986). Kintsch and van Dijk (1978) study the mental operations underlying the comprehension and summarization of text. They introduce *propositions* as the basic unit of text representation, and a model of how incoming text is processed given memory limitations; texts are reduced to important propositions (to be recalled later) using *macro-operators* (e.g., addition, deletion). Their model has met with popularity in cognitive psychology (Baddeley, 2007) and has also found application in summarization (Fang and Teufel, 2016).

Our work proposes a new encoder for contextualized semantic parsing. At the heart of our approach is a memory controller which keeps track of context via writing new information and updating old information. Our memory-based approach is inspired by Kintsch and van Dijk (1978) and is closest to Santoro et al. (2016), who use a memory augmented neural network (Weston et al., 2015; Sukhbaatar et al., 2015) for meta-learning. Specifically, they introduce a method for accessing external memory which functions as short-term storage for meta-learning. Although we report experiments solely on semantic parsing, our encoder is fairly general and could be applied to other context-dependent tasks such as conversational information seeking (Dalton et al., 2020) and information retrieval (Sun and Chai, 2007b; Voorhees, 2004).

3.7 Summary

In this chapter, we presented a memory-based model for context-dependent semantic parsing and evaluated its performance on a text-to-SQL task. We hypothesized that *memory-based representations of discourse are effective in long-range interactive semantic parsing*. Our discourse memory is represented as a matrix that as the interaction proceeds maintains relevant information required to response to the user utterance. Analysis of model output revealed that our approach is able to handle several discourse related phenomena to a large extent. We also analyzed the behavior of the memory controller and observed that it correlates with the model's output decisions. Our study indicates that explicitly modeling context can be helpful for contextual language processing tasks.

In this chapter, we show that bounded memory is effective in maintaining a cumulative meaning of discourse which partly validates our primary hypothesis of this thesis namely that *maintaining explicit representations of context (external or discourse) are helpful for tasks requiring long-range interactions*. However the work in this chapter assumes

that the external context (a database schema in that case) can be fully encoded by the model. This assumption does not hold for tasks where the external context is very large, like a full knowledge graph (KG). A KG, such as Wikidata, can have millions of entities and thousands of relations; and therefore cannot be encoded in its entirety by a model. In Chapters 4 and 5, we will move away from this assumption and show that dynamic graphs can effectively represent large external knowledge graph contexts without needing to encode the entire context simultaneously.

Chapter 4

Semantic Parsing for Conversational Question Answering over Knowledge Graphs

In the previous chapter, we introduced and discussed a memory-based model designed for conversational semantic parsing with a relational database as *external* context. We validated our text-to-SQL semantic parser on three different datasets, namely ATIS (Suhr et al., 2018), SPARC (Yu et al., 2019b), and COSQL (Yu et al., 2019a). Our findings supported the hypothesis that *memory-based representations of discourse are effective in long-range interactive semantic parsing*, showing promising results in handling various discourse-related phenomena. Further inspection of our learned memory controller confirmed its significant role in influencing the model's output decisions, reinforcing the utility of modeling past context for long-range language processing tasks.

Our memory-based approach in Chapter 3 focused solely on modeling discourse context, operating under the assumption that all external context could be fully encoded by the model. This assumption holds for relatively small external contexts, such as a database schema in that case. In this chapter, we extend our investigation and include larger external contexts such as the Wikidata knowledge graph.

We begin this chapter by introducing $\mathbb{SPICE}$, a new dataset is designed for conversational semantic parsing tasks requiring long interactions with Wikidata. We present two models for this task that employ different modeling techniques. LasagneSP (Kacupaj et al., 2021) utilizes a coarse-to-fine (Dong and Lapata, 2018) approach, while BertSP implements direct semantic parse generation using dynamic vocabulary (Gu et al., 2021). In the next Chapter (5), we will improve upon methods presented in the chapter by

modeling context explicitly and efficiently in the form of dynamic graphs.

SPICE stands for **S**emantic **P**ars**I**ng dataset for **C**onversational qu**E**stion answering over Wikidata. SPICE consists of user-assistant interactions where natural language questions are paired with SPARQL parses, and answers provided by the system correspond to SPARQL execution results. We derive this dataset from CSQA (Saha et al., 2018b), an existing benchmark originally proposed for retrieval-based conversational question answering (Lan et al., 2021). The benchmark uses Wikidata as the underlying knowledge graph. Wikidata provides a large scale, openly accessible, and continuously updated multilingual knowledge graph (Schmelzeisen et al., 2021). It supports SPARQL, a query language specifically designed for retrieving data from the RDF[1]) format [2], which is the format Wikidata uses to store its information. This allows users to access and query the data in a structured and precise way. Wikidata snapshot *14-Nov-2016* for the developement of the dataset which contains 5.2K relations, 12.8M entities and 52.3M facts. Although CSQA does not have executable queries, it contains a large number of natural language questions and their corresponding answers, highlighting a range of conversational phenomena such as coreference, ellipsis, and topic change as well as different types of questions exemplifying varying intents.

Figure 4.1 shows a conversation from SPICE illustrating how questions (utterances on the left) are annotated with SPARQL queries (on the right blue box). To create a large-scale dataset (197k conversations), we develop SPARQL templates for different question intents; entity, relation, and class symbols are initially under-specified and subsequently filled automatically to generate full SPARQL queries. CSQA questions have been previously associated with logical forms generated with custom-made grammars (Guo et al., 2018; Kacupaj et al., 2021; Marion et al., 2021). As a result, semantic parsers based on them operate with different sets of grammar rules and are not strictly comparable, since the grammars may have different coverage and semantics (e.g., terminal symbols may encapsulate different degrees of execution complexity). In SPICE, questions are represented with SPARQL[3], a standard query language for retrieving and manipulating RDF data. This allows us to compare parsers developed on the dataset on an equal footing and facilitates further extensions (e.g., new question intents), without the need to redefine the grammar and its execution engine. In an attempt to build semantic parsers which generalize to new entities and concepts, we further create different data

[1]RDF (Resource Description Framework represents information as triples of subject-predicate-object, allowing for structured data representation and interoperability across different systems.

[2]https://www.w3.org/TR/PR-rdf-syntax/Overview.html

[3]https://www.w3.org/TR/sparql11-query/

	Utterances	Annotations	Actions and Semantic Parses
$\mathcal{T}_1$	U: Which tournament did Detroit Tigers participate in? S: 1909 World Series	INTENT = Simple Question | Single Entity ENT = Q650855 (Detroit Tigers) REL = P1923 (participating team) TYPE = Q500834 (tournament) TRIPLE = (Q500834,P1923,Q650855) GOLD = Q846847 (1909 World Series)	Action Sequence: [filter_type,find_rev,Q650855,P1923,Q500834] Sparql Parse: SELECT ?x WHERE { ?x wdt:P1923 wd:Q650855. ?x wdt:P31 wd:Q500834.}
$\mathcal{T}_2$	U: Which sports team was the champion of that tournament? S: Pittsburgh Pirates	INTENT = Simple Question | Single Entity | Indirect ENT = Q846847 (1909 World Series) REL = P1346 (winner) TYPE = Q12973014 (sports team) TRIPLE = (Q846847,P1346,Q12973014) GOLD = Q7199360 (Pittsburgh Pirates)	Action Sequence: [filter type, find, Q846847, P1346, Q12973014] Sparql Parse: SELECT ?x WHERE { wd:Q846847 wdt:P1346 ?x. ?x wdt:P31 wd:Q12973014.}
$\mathcal{T}_3$	U: Does that sports team belong to Sacile? S: No	INTENT = Verification | 2 entities, subject is indirect ENT = [Q653772 (Pittsburgh Pirates), Q53190 (Sacile)] REL = P17 (country) TYPE = Q15617994 (designation admin. territorial entity) TRIPLE = (Q653772,P17,Q53190) GOLD = False	Action Sequence: [filter type, find, Q846847, P1346, Q12973014] Sparql Parse: SELECT ?x WHERE { wd:Q846847 wdt:P1346 ?x. ?x wdt:P31 wd:Q12973014.}

Figure 4.1: Example conversations from SPICE. The left column shows dialogue turns ($\mathcal{T}_1$–$\mathcal{T}_3$) with user (U) and system (S) utterances. The middle column shows the annotations provided in CSQA. Boxes on the right show the sequence of actions and corresponding SPARQL semantic parses (SP).

splits where new intents appear only at test time (Finegan-Dollak et al., 2018).

For our semantic parsing task, we establish two strong baseline models which tackle the large vocabulary problem and the prediction of logical forms in different ways. The first approach (Gu et al., 2021) uses *dynamic vocabularies* derived from KG subgraphs for each question and a simple sequence-to-sequence architecture to predict *complete* SPARQL queries. The other approach (Kacupaj et al., 2021) predicts SPARQL *query templates* and then fills in entity, relation, and type slots by means of an entity and ontology classifier. Our experiments reveal several shortcomings in both approaches, such as not being able to encode large sets of KG elements and generate the same entity several times. Both approaches struggle with ellipsis, they cannot resolve coreference when the referent appears in the conversation context beyond the previous turn, have reduced performance on questions with multiple entities, and difficulty with unseen question intents. We explore these challenges and outline research directions for conversational semantic parsing, which we address in Chapter 5.

4.1 The SPICE Dataset

The CSQA dataset (Saha et al., 2018b) aims to facilitate the development of QA systems that handle complex and inter-related questions over a knowledge graph (KG). In contrast to simple factual questions that can be answered with a single KG triple (i.e., {subject, relation, object}), complex questions require manipulating sets of triples and reasoning over these. In Figure 4.1, a question like *How many sports teams participated in that tournament?* in turn $\mathcal{T}_2$ requires numerical reasoning; it also relies on correctly interpreting $\mathcal{T}_1$.

Questions and answers in this dataset were elicited from human experts playing user and system roles as well as from crowdworkers. In a second stage, templates derived from the human-authored QA pairs were used to automatically augment the dataset. Human experts also suggested complex reasoning questions and derived templates thereof. Conversations were built as sequences of QA pairs exploring paths in the KG. By construction, the QA pairs in a conversation are connected through one or several entities in the KG. Questions fall into two coarse categories, *simple* and *reasoning*-based, and the way QA pairs are organised in a sequence introduces various *conversational phenomena* which we summarize below.

Simple Questions are factoid questions, seeking information related to an entity (e.g., *Which tournament did Detroit Tigers participate in?* in Figure 4.1) or set of entities (e.g., *What are the countries of those sports teams?*).

Reasoning Questions are complex questions which require the application of numerical and logical operators over sets of entities. For instance, to answer the question *How many sports teams participated in that tournament?* requires finding the set of sports teams that participated in a given tournament (e.g., *1909 World Series*) and taking its count. Questions in this category also involve General Entities (GE) such as *tournament*, in addition to Named Entities (NE), and multiple entities (both NE and GE) in a single question (e.g., *Which tournaments have less number of participating sports teams than 1909 World Series?*). Some question types also combine multiple reasoning operators.

Conversations contain sequences of mixed-initiative interactions where the system requests clarification on ambiguous questions. Conversations also include discourse phenomena such as coreference (e.g., *Which sports team was the champion of that*

Nb. instances	197K
Nb. entities	12.8M
Nb. relations	2738
Nb. types	3064
Average turn length	9.5
Average entities per conversation	7.6
Average types per conversations	6.5
Average neighbourhood per turn	181.4 triples
Logical Reasoning, Quantitative Reasoning, Comparative Reasoning, Quantitative Reasoning Count, Comparative Reasoning Count, Verification, Simple Question	
Clarification, Coreference, Ellipsis	

Table 4.1: Overall statistics of SPICE dataset (top); general question types (middle); linguistic phenomena (bottom).

tournament? in Figure 4.1) and ellipsis (e.g., *And what about 1910 World Series?* as a follow up question for *How many sports teams participated in that tournament?*).

4.1.1 Question Semantics Described by Actions

Saha et al. (2018b) envisaged CSQA as a benchmark for retrieval-based conversational question answering (Bordes et al., 2015; Dong et al., 2015; Jain, 2016; Lan et al., 2021). These methods embed natural language questions and KG triples into high dimensional spaces and rely on neural reasoning modules to match questions to candidate answers. Hence, questions do not have associated logical forms, only gold answers are available.

Our success in creating semantic parse annotations is partly due to the fact that CSQA provides useful KG information. Each interaction (i.e., user and system turn) comes with annotations about KG entities, types, and relation symbols as well as some information about the triple patterns involved in the question.

As illustrated in Figure 4.1 for question *Which tournament did Detroit Tigers participate in?* annotations contains with ENT=Q650855 (Detroit Tigers), REL=P1923 (participating team), TYPE=Q500834 (tournament, and TRIPLE=(Q500834,P1923,Q650855) fields. It also provides information pertaining to question types and sub-types (see INTENT=Simple Question|Single Entity in Figure 4.1).

Taking advantage of these annotations, follow-on work (Guo et al., 2018) defined a

	ATIS	SParC	CoSQL	SPICE
Nb. Instances	1,658	4,298	3,007	197K
Average turn length	7.0	3.0	5.2	9.5
Domain	Single	Multiple	Multiple	Wikidata
Logical form	SQL	SQL	SQL	SPARQL
Database type	DB	DB	DB	KG

Table 4.2: Comparison with other related datasets for conversational semantic parsing (DB: relational database; KG: knowledge graph). Datasets other than SPICE are over relational database external context with shorted interaction length.

semantic parsing task over CSQA, modeling the meaning of questions as a sequence of actions. The set of actions encompasses find (or find_rev when the entity is in object position) to retrieve sets of entities in a subject (object) position, as well as actions operating on sets of entities (e.g., filter_type). For instance, the question *Which tournament did Detroit Tigers participate in?* in turn $\mathcal{T}_1$ in Figure 4.1 would be parsed to [filter_type, find_rev, Q650855, P1923, Q500834], meaning "find the set of entities that are in relationship *participating team* with *Detroit Tigers* and then filter those that are of type *tournament*". A breadth-first search algorithm generates action-grammar annotations for each question and a sequence of grammar-actions is considered correct if upon execution it returns the gold answer. Subsequent work (Shen et al., 2019; Kacupaj et al., 2021; Marion et al., 2021) expanded this action-grammar greatly improving its coverage (i.e., the number of successfully annotated questions).

4.1.2 From Actions to SPARQL Queries

In this work, we take a step further and map CSQA natural language questions into vanilla SPARQL queries. We first analysed how intent is expressed in question types and sub-types and then manually defined SPARQL templates for each question sub-type. A SPARQL template is a query with unspecified triple patterns in the WHERE clause. For instance, the template for the question in turn $\mathcal{T}_1$ is {SELECT ?x WHERE ?x RELATION ENTITY. ?x wdt:P31 TYPE.}. We finally modified the tool provided in Kacupaj et al. (2021) to automatically instantiate the SPARQL templates, providing annotations for the entire dataset (e.g., by filling missing slots as SELECT ?x WHERE {?x wdt:P1923 wd:Q650855. ?x wdt:P31 wd:Q500834.}).

We imported the Wikidata snapshot provided by Saha et al. (2018b) into a KG in a SPARQL server (see Section 4.3 for more details) and assessed the correctness of SPARQL queries by executing them and comparing results to gold answers. For some questions the annotation procedure did not produce a SPARQL parse that recovered the gold answer. In these rare cases, we redefined the answer if it did not affect the conversation flow or truncated the conversation up to that point.

Table 4.1 shows various statistics for SPICE while Table 4.2 compares it to related conversational datasets such as ATIS (Suhr et al., 2018), SParC (Yu et al., 2019b), and CoSQL (Yu et al., 2019a). As can be seen, SPICE contains a sizeable number of training instances, its conversations are longer, and the semantic parsing task is real-scale. In Table 4.3, provides the list of question along with their sub-types. For each question sub-type we provide an example user question. For cases involving ellipsis and coreference, we include the conversation context.

Table 4.3: Full list of question sub-types (intents) in SPICE. Each sub-type includes an example user question, particularly when it involves conversational phenomena such as coreference or ellipsis. Min/Max queries seek the smallest or largest values, while At least/At most set thresholds for minimum and maximum results. Approx. queries involve conditions within a close range. More/Less compares counts against specific values, and Count requests the total number of matching results. Set operations, Difference finds set difference between two sets, Intersection identifies common results between sets, and Union combines results from multiple sets into a single set. Previous conversational interactions necessary for interpreting the user question are shown in grey.

Question Sub-Type	Example User Question
Clarification	
Simple Question — Single Entity (Coreference)	U: *Which political territory is that sporting event located in?* S: *Did you mean Speed skating at the 2010 Winter Olympics – Men's 500 metres?* U: *Yes*
Logical Reasoning	
Difference — Multiple Relation	U: *Which people were awarded with Order of Merit for Arts and Science and are not working as singer?*
Difference — Single Relation	U: *Which international organizations had Poland but not Bulgaria as their member?*
Difference — Single Relation (Ellipsis)	U: *Which city was Pierre Laffont born in?* S: *Marseille* U: *Which administrative territories are the sister cities of that city?*

Continued on next page

Table 4.3 – *Continued from previous page*

Question Type/Sub-Type	Example User Question
	S: *Shanghai, Odessa, Naples* U: *But not Bologna*
Intersection — Multiple Relation	U: *Which human settlements are situated close to Trave and have an adjacent border with Herzogtum Lauenburg?*
Intersection — Single Relation	U: *Which works of art were filmed at Edinburgh and Berlin?*
Intersection — Single Relation (Ellipsis)	U: *Which language does José María Lassalle speak in?* S: *Spanish* U: *And also Sergio Gil*
Union — Multiple Relation	U: *Which watercourses are located in the neighbourhood of Bremen or are the tributaries of Ob?*
Union — Single Relation	U: *Which people are the creators of The Theory of Everything or Ten Minutes to Live?*
Union — Single Relation (Ellipsis)	U: *What is the profession of Mai Yamada?* S: *announcer* U: *Or Kazimierz Rogoyski?*
Quantitative Reasoning	
Min/Max — Single entity type	U: *Which musical instruments are played by min number of people?*
Min/Max — Mult. entity type	U: *Which organizations are the main building contractors of max number of architectural structures and buildings?*
Atleast/ Atmost/ Approx. the same/Equal — Single entity type	U: *Which musical instruments are played by exactly 5 people?*
Atleast/ Atmost/ Approx. the same/Equal — Mult. entity type	U: *Which events are demonstrated in atleast 3 prints and genres of sculpture?*
Comparative Reasoning	
More/Less — Mult. entity type	U: *Which landforms are known for containing lesser number of chemical compounds or minerals naturally than Stetind pegmatite?*
More/Less — Mult. entity type (Ellipsis)	U: *Which landforms are known for containing lesser number of chemical compounds or minerals naturally than Stetind pegmatite?* S: *Euboea, Izalco, Mount Nyiragongo* U: *And also tell me about Tuften quarry?*
More/Less — Mult. entity type (Coreference)	U: *Which administrative territory is that person a civilian of?* S: *Spain* U: *Which administrative territories are the countries of origin of lesser number of television programs or works of art than that administrative territory?*
More/Less — Single entity type	U: *Which television programs have been dubbed by more number of people than Puss in Boots: The Three Diablos?*
More/Less — Single entity type (Ellipsis)	U: *Which television programs have been dubbed by more number of people than Puss in Boots: The Three Diablos?*

Continued on next page

Table 4.3 – *Continued from previous page*

Question Type/Sub-Type	Example User Question
	S: *House, K-On!, K-On!!* U: *And also tell me about Chip 'n Dale Rescue Rangers?*
More/Less — Single entity type (Coreference)	U: *Which languages are max number of literary works composed in?* S: *English* U: *Which languages are the mother tongues of less number of people than that language?*
Simple Question (Direct)	
Simple Question	U: *Which type of sport did Amel Tuka participate in?*
Single Entity	U: *What is the capital of Sweden?*
Mult. Entity	U: *Who were the writers of On being and essence, De vegetabilis et plantis libri septem and Historia de regibus Gothorum, Vandalorum et Suevorum?*
Simple Question (Ellipsis)	
only subject is changed, parent and relation remains same	U: *Which organizations are the sponsors of Janice Anderson?* S: *Montrail, Patagonia, Inc.* U: *And also tell me about Manikala Rai?*
object parent is changed, subject and relation remain same	U: *Which watercourses are situated nearby Munich?* S: *Eisbach, Würm, Isar* U: *And which river?*
Simple Question (Coreference)	
Mult. Entity	U: *Which releases have Motown as their record label?* S: *What's Going On, Got to Be There, Can't Slow Down* U: *Which genre do those releases belong to?*
Single Entity (Coreference)	U: *Which narrative location is The Penalty set in?* S: *San Francisco* U: *Which color is associated with that film genre?*
Verification (Boolean)	
2 entities, both direct	U: *Is Zugspitze located in Germany?*
2 entities, one direct and one corefered, object is corefered	U: *Which university was Eden Stiles educated at?* S: *University of Michigan* U: *And what about C. V. Raman?* S: *University of Madras* U: *Was Ravindra Wijegunaratne educated at that university?*
2 entities, one direct and one corefered, subject is corefered	U: *Which German business organization was Gustav Peichl a member of?* S: *Academy of Arts, Berlin* U: *What was designed by that person?* S: *Millennium Tower* U: *Does that tower block belong to Austria?*
3 entities, 2 direct, 2(direct) are query entities, subject is corefered	U: *Which administrative territory was Gary Collier born at?* S: *Fort Worth*

Continued on next page

Table 4.3 – *Continued from previous page*

Question Type/Sub-Type	Example User Question
	U: *Is that administrative territory a sister city of Adamsville, New Brunswick and Yuen Long Kau Hui?*
3 entities, all direct, 2 are query entities	U: *Is Aix-en-Provence partner town of Baton Rouge and Hemmatabad, Alborz?*
one entity, multiple entities (as object) coreferred	U: *Which armed conflicts are Battle of the Argeş or Battle of the Yellow Sea a part of?* S: *Romania during World War I, Russo-Japanese War* U: *Did those armed conflicts fight in Rui Natsukawa?*
Quantitative Reasoning (Count)	
Incomplete count-based ques	U: *How many people influenced Chris Marker?* S: *1* U: *And also tell me about Ada Yonath?* S: *1* U: *And what about Mikhail Bakunin?*
Count over Atleast/ Atmost/ Approx. the same/Equal—Mult. entity type	U: *How many cities are the terminus locations of atleast 5 thoroughfares and roads?*
Count over Atleast/ Atmost/ Approx. the same/Equal—Single entity type	U: *How many musical instruments are played by exactly 2 people?*
Count — Logical operators	U: *How many bodies of water or watercourses are situated nearby Lübeck?*
Count — Logical operators (Coreference)	U: *Which administrative territory is the native country of Carolina Goic Boroevic?* S: *Chile* U: *Who is the head of the government of that administrative territory?* S: *Michelle Bachelet* U: *What is the capital of that administrative territory?* S: *Santiago* U: *How many capitals or cities are sister towns of that city?*
Count — Mult. entity type	U: *How many people starred in Django Kill or Shatterday?*
Count — Single entity type	U: *How many people starred in Captain America: Civil War?*
Count — Single entity type (Coreference)	U: *Which armed conflict did Lionel of Antwerp, 1st Duke of Clarence take part in?* S: *Hundred Years' War* U: *How many people did that armed conflict engage in?*
Comparative Reasoning (Count)	
Count over More/Less — Mult. entity type	U: *How many administrative territories have adopted lesser number of holidays and people as patron saint than Santo Stefano al Mare?*
Count over More/Less — Mult. entity type (Ellipsis)	U: *How many administrative territories have adopted lesser number of holidays and people as patron saint than Santo Stefano al Mare?* S: *296* U: *And what about San Donato Milanese?*

Continued on next page

Table 4.3 – *Continued from previous page*

Question Type/Sub-Type	Example User Question
Count over More/Less — Mult. entity type (Coreference)	U: *Which administrative territories are Luigi Einaudi the head of state of and have UTC+01:00 as their time zone?* S: *Italy* U: *How many administrative territories are the origins of greater number of literary works or releases than that administrative territory?*
Count over More/Less — Single entity type	U: *How many legislatures represent lesser number of states than East Bengal Legislative Assembly?*
Count over More/Less — Single entity type (Ellipsis)	U: *How many legislatures represent lesser number of states than East Bengal Legislative Assembly?* U: *207* U: *And how about Estates of Curaçao?*
Count over More/Less — Single entity type (Coreference)	U: *Which french administrative division was Philippe Esnault born in?* S: *Alençon* U: *Which occupation has that person as his/her 's career?* S: *historian* U: *Which administrative territory is the native country of that person?* S: *France* U: *How many administrative territories inspired less number of fictional locations than that administrative territory?*

4.2 Problem Formulation

We recall the problem formulation for conversational semantic parsing from Chapter 3. Different from the previous chapter where we worked with relational databases as external context, in this chapter, we will focus on Wikidata knowledge graph $\mathcal{K}$ as the external context. We denote a sequence of dialogue turns in an interaction as $I = [X_t, Z_t, A_t]_{t=1}^{T}$. Here X_t represents a user's utterance, Z_t is the semantic parse, and each A_t is the corresponding answer at turn t. Given previous *discourse* context C_t derived from $I[:t-1]$ and current user question $X_t = (x_{t1}, x_{t2}, \cdots, x_{t|X_t|})$, our goal is to predict a SPARQL query $Z_t = (z_{t1}, z_{t2}, \cdots, z_{t|Z_t|})$ that represents the intent of X_t and, upon execution over knowledge-graph $\mathcal{K}$, yields denotation A_t. Z_t is a sequence over a target vocabulary $\mathcal{V} = \mathcal{V}_f \cup \mathcal{V}_{\mathcal{K}}$ where $\mathcal{V}_f$ is fixed and contains SPARQL keywords (e.g., SELECT) and special tokens (e.g., beginning of sequence token, BOS), and $\mathcal{V}_{\mathcal{K}}$ contains all knowledge-graph symbols (e.g., entity IDs such as Q76 for *Barack Obama*).

We propose two approaches for this semantic parsing task which establish strong baseline performance and highlight various challenges. These differ in the way they handle large KG vocabularies and how they generate logical forms. Figure 4.2 and 4.3, provides a sketch of the two models discussed in the subsequent sections.

4.2.1 Parsing with a Single Decoder and Knowledge subgraphs

Figure 4.2 shows the overall architecture of our first model. Our model is parameterised by an encoder-decoder transformer neural network (Vaswani et al., 2017b). Our encoder uses a BERT as base model to contextualize and encode both user utterances and relevant KG items. Meanwhile, our decoder is a modified transformer decoder designed to handle a dynamic vocabulary required for KG items, which varies depending on the current question. Next, we will describe the process of extracting question dependent KG vocabulary, followed by a detailed explanation of our encoder-decoder model.

Dynamic Vocabulary Since the KG vocabulary $\mathcal{V}_{\mathcal{K}}$ can be extremely large, we parse question X_t with a smaller vocabulary $\mathcal{V}_t \subseteq \mathcal{V}_{\mathcal{K}}$ which only contains KG symbols related to X_t. Following previous work (Gu et al., 2021; Marion et al., 2021), we assume the symbols related to X_t are those appearing in subgraph $\mathcal{G}_t$ of knowledge-graph $\mathcal{K}$, $\mathcal{G}_t \subseteq \mathcal{K}$. Given question X_t and its discourse context C_t (that is a concatenation of previous utterance-response pairs $I[:t-1]$), we identify KG entities $\mathcal{E}_t = \{e_{t1}, e_{t2}, \cdots, e_{t|\mathcal{E}_t|}\}$ which correspond to mentions in X_t and C_t. In Section 4.5.1, we will discuss different variants of performing entity recognition and linking that provides us KG entities $\mathcal{E}_t$. We then obtain $\mathcal{G}_t$ by taking the one-hop neighbourhood for each entity $e_{ti} \in \mathcal{E}_t$. In other words, we include all KG triples (s, r, o) where the entity appears in subject ($s = e_{ti}$) or object position ($o = e_{ti}$). When e_{ti} is a subject, we include triple (e_{ti}, r, τ_o) where τ_o is the type of entity o; analogously, when e_{ti} appears in an object position, we add (τ_s, r, e_{ti}). For entities e_{ti} we include their types $\tau_{e_{ti}}$. When e_{ti} is a general entity (e.g., a type such as *tournament*) we add relations from $\mathcal{K}$ that have instances of type e_{ti} as their subject (object). The final vocabulary $\mathcal{V}_t$ contains all entities in $\mathcal{E}_t$, all relations r and types (τ_o, τ_s, and $\tau_{e_{ti}}$) found in the set of triples in $\mathcal{G}_t$. Following previous work (Marion et al., 2021; Kacupaj et al., 2021), we restrict the discourse context only to the previous user-system interaction $I[t-1]$.

Encoder-Decoder Model Our encoder is a BERT (Devlin et al., 2019) model fine-

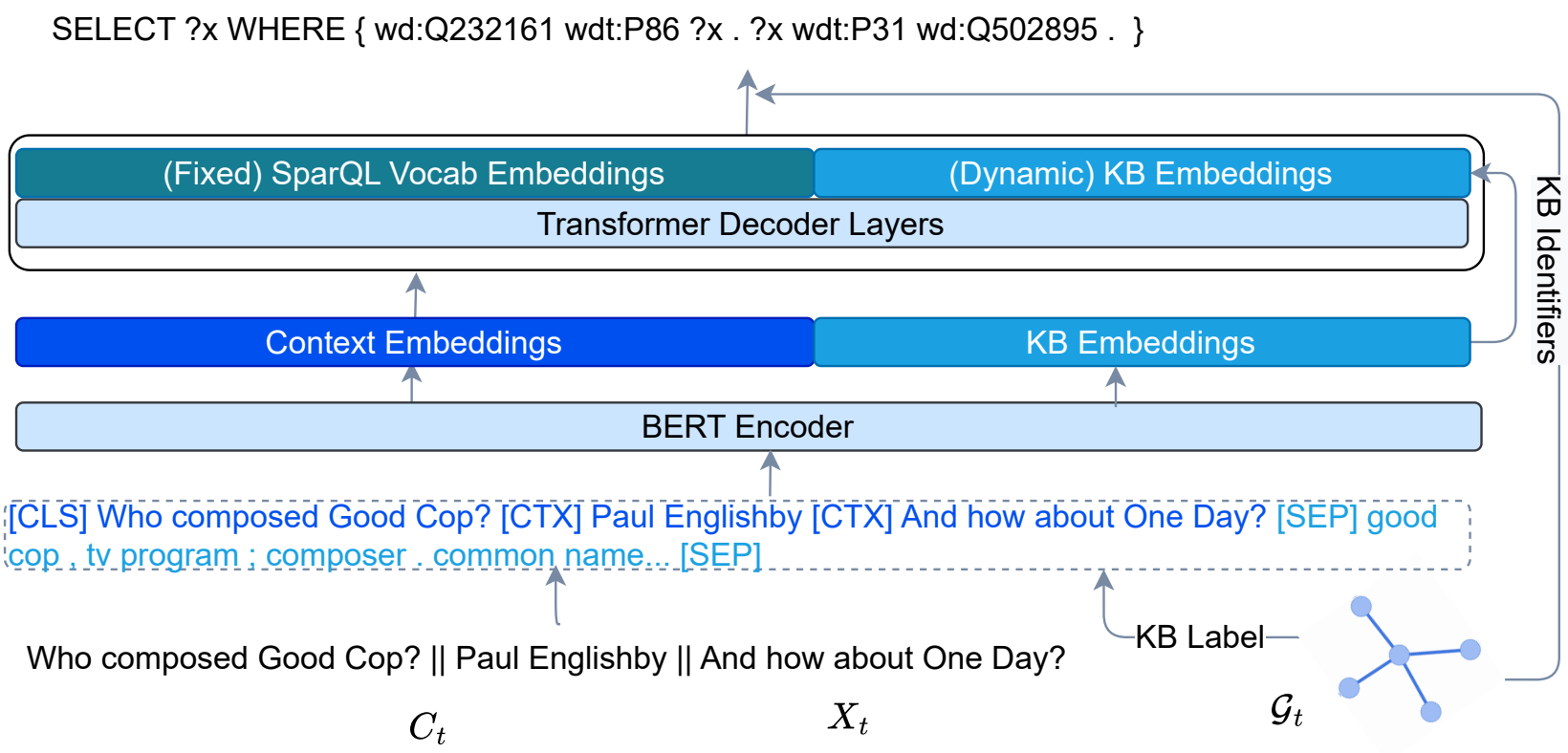

Figure 4.2: Our proposed BertSP model. BERT-based encoder for contextualizing the current utterance X_t (*And how...*), previous *discourse* context C_t (*Who composed...*), and the relevant KG items (composer) that are retrieved from *external* context. Upper part of the figure shows modified transformer decoder for handling dynamic KG vocabularies. The decoder maps verbalized KG items back to their identifiers at prediction using subgraph $\mathcal{G}_t$, such as composer $\Rightarrow$ P86.

tuned on our semantic parsing task. The decoder is a randomly initialised Transformer network (Vaswani et al., 2017b). To account for the difference in initialisation between the encoder and decoder networks, we follow the training scheme proposed in Liu and Lapata (2019). We provide details in Section 4.4.

The input to our semantic parser is a tuple $(X_t, C_t, \mathcal{G}_t)$ consisting of natural language question X_t, its context C_t, and subgraph $\mathcal{G}_t$ which we adapt to BERT's input format as follows (Gu et al., 2021). We concatenate the sequence of natural language questions X_{t-1} and X_t, using the special token [CTX] as a delimiter and prepend the [CLS] token in the beginning of the sequence. Special token [SEP] denotes the end of sequence followed by the linearized KG subgraph $\mathcal{G}_t$. The linearisation procedure goes over entities in $\mathcal{G}_t$, enumerating their types and relations. Importantly, we denote entities by their label rather than their KG identifiers. The order of entities in $\mathcal{G}_t$ is random. Figure 4.2 shows an example of the input to our BERT-based encoder.

More formally, the encoder takes token sequences $X'_t = \text{[CLS]}X'_{t_{\text{text}}}\text{[SEP]}X'_{t_{\text{graph}}}\text{[SEP]}$ as input where $X'_{t_{\text{text}}}$ is the natural language sequence that represents the *discourse* context C_t (*Who composed...* in Figure 4.2) along with current utterance X_t (*And how...*). While $X'_{t_{\text{graph}}} = (g^t_1, \cdots, g^t_{|\mathcal{G}_t|})$ is the *external* context represented as the sequence of knowledge-graph symbols from the linearized graph $\mathcal{G}_t$. Note that these knowledge-graph symbols constitute the target dynamic vocabulary $\mathcal{V}_t$ and $|\mathcal{G}_t|$ represents the number of KG symbols which is equal to the size of the target vocabulary $|\mathcal{V}_t|$. The encoder maps input sequences X'_t into sequences of continuous representations $\mathbf{z_t} = (\mathbf{z}_{t1}, ..., \mathbf{z}_{t|X_t|})$, and the decoder then generates the target SPARQL parse $Z_t = (z_{t1}, ..., z_{t|y_t|})$ token-by-token autoregressively, hence modeling the conditional probability: $p(z_{t1}, ..., z_{t|Z_t|} \,|\, X'_t)$. The linearized graph $\mathcal{G}_t$ can exceed BERT's maximum number of input positions (which is 512). To avoid throwing away useful information, we adopt a solution similar to Gu et al. (2021). For question X_t with $\mathcal{G}_t$ containing k entities, we create k input sequences $X'^1_t, \cdots, X'^k_t$. These k sequences share the natural language subsequence but have different KG symbol subsequences. Given an input sequence $x'^1_t, \cdots, x'^k_t$, we obtain contextualised representations as $\mathbf{z}^1_t, \cdots, \mathbf{z}^k_t = \text{BERT}(x'^1_t, \cdots, x'^k_t)$.

The model further splits the sequence of continuous representations $\mathbf{z}^j_t$ into textual representations $\mathbf{z}^j_{t_{\text{text}}}$ and knowledge-graph symbols $\mathbf{z}^j_{t_{\text{graph}}}$ both of which are contextualised. We then average representations $\mathbf{z}_{t_{\text{text}}} = \text{AVG}(\mathbf{z}^j_{t_{\text{text}}})$ and feed them as input to the decoder (see Figure 4.2). From representations $\mathbf{z}^j_{t_{\text{graph}}}$, we derive the embeddings for the elements in the target dynamic vocabulary $\mathcal{V}_t$. Recall that the decoder parses input questions x_t using target vocabulary $\mathcal{V} = \mathcal{V}_f \cup \mathcal{V}_t$ which consists of a set of fixed

($\mathcal{V}_f$) and dynamic ($\mathcal{V}_t$) target tokens. The decoder then predicts the probability of each SPARQL token z_{ti} as $p(z_{t_i}|z_{t_{<i}}, x_t'^1, \cdots, x_t'^k) = \text{softmax}(\mathbf{W}_o \mathbf{h}_i^L)$ where $\mathbf{h}_i^L$ is the decoder top layer hidden representation at time step i. $\mathbf{W}_o \in \mathbb{R}^{|\mathcal{V}_f \cup \mathcal{V}_t|}$ is the output embedding matrix with $\mathbf{W}_o = [\mathbf{W}_f; \mathbf{W}_t]$, where $[;]$ denotes matrix concatenation, and $\mathbf{W}_t$ is derived from the encoder representations $\mathbf{z}^j_{t_{\text{graph}}}$.

4.2.2 Parsing with Multiple Decoders and an Ontology Classifier

Our second model is an adaptation of the Lasagne architecture proposed in Kacupaj et al. (2021) for our task. Figure 4.3 shows the overall architecture. Lasagne generates logical forms following a multi-stage approach where a backbone sketch is first predicted and then fleshed out. Their sketch is a sequence of actions from a custom grammar which we modify to be a sketch of SPARQL queries.

SPARQL Template Prediction Lasagne employs an encoder-decoder model based on Transformers (Vaswani et al., 2017b) to convert a user question X_t in a conversation into a logical form template. The input to the encoder is the discourse context C_t and user question X_t. Utterances are separated via [SEP] tokens, while the special context token [CTX] denotes the end of sequence (see Figure 4.3). The input sequence is encoded via multi-head attention (Vaswani et al., 2017b) to output contextualized representations which are then fed to the decoder to predict a sequence of actions (without grounding to KG elements) token-by-token. Instead, our decoder predicts SPARQL queries with place-holders for KG symbols. For instance, for the WHERE clause of turn $\mathcal{T}_1$ in Figure 4.1, it predicts {ENTITY RELATION ?x. ?x wdt:P31 TYPE.} instead of {wd:Q5582479 wdt:P161 ?x. ?x wdt:P31 wd:Q502895.})

Entity Recognition and Linking An entity recognition module detects entities in the input and links them to the KG (Shen et al., 2019; Kacupaj et al., 2021). Initially, entity spans are identified using an LSTM which performs BIO sequence labelling.[4] Entity spans are subsequently linked to KG entities via an inverted index (created using Elasticsearch[5]) which maps entity labels to entity IDs. Once identified, the entities are further filtered and reordered so that that they match their order of appearance in the SPARQL (see Figure 4.3).

[4] BIO labels for training are obtained by preforming string matching between entity annotations and user utterances.

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

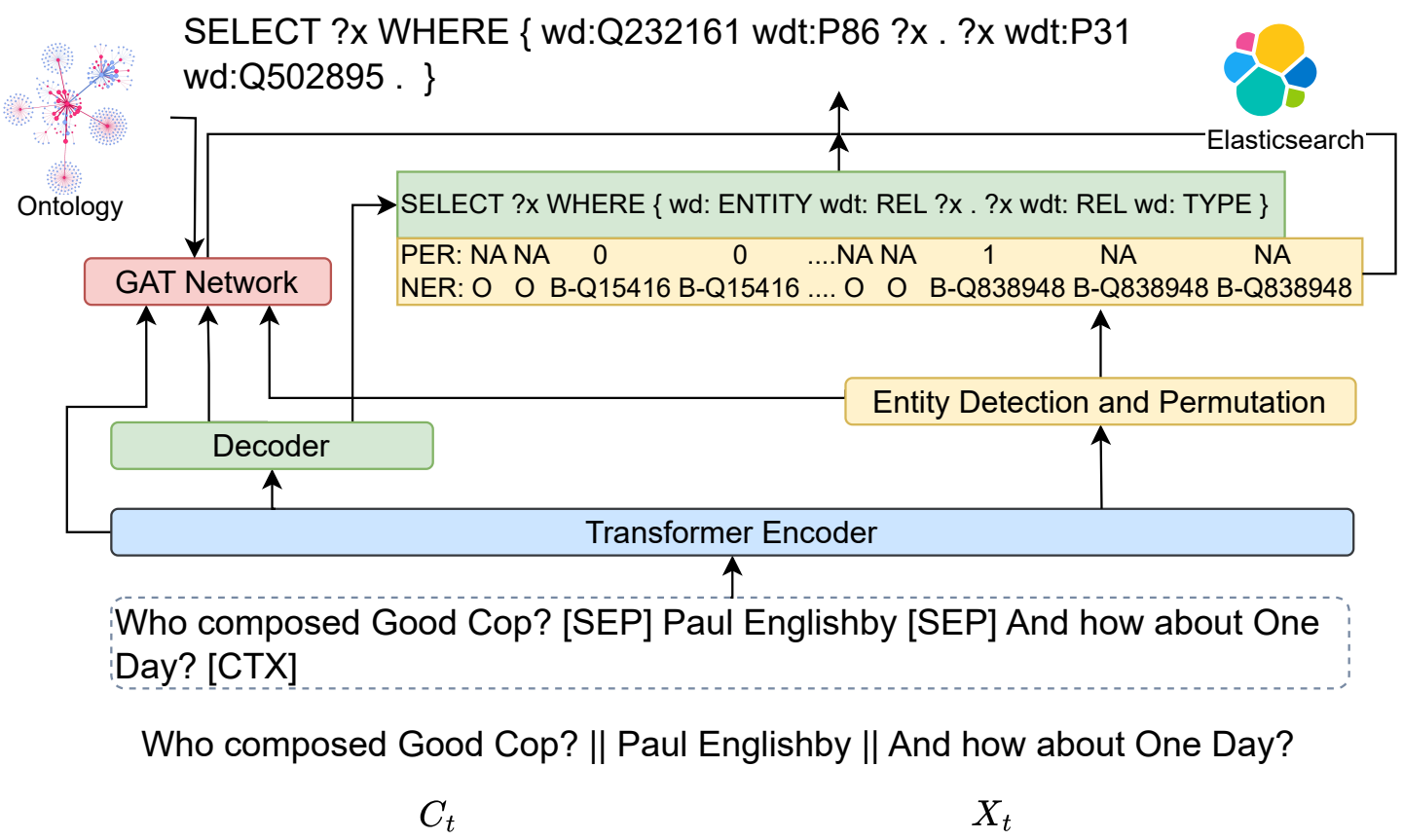

Figure 4.3: Overview of our sketch-based semantic parsing model. It employs a transformer based encoder to contextualize previous discourse context C_t (*Who composed...*) with the current utterance X_t (*And how...*). It initially generates a SPARQL sketch with placeholders for KG elements. These placeholders are filled using an ontology graph representation through a GAT Network, complemented by an entity recognition and linking module using BIO sequence labeling.

Predicting Types and Relations Finally, an ontology graph with types and relations appearing in SPICE's KG is constructed.[6] The graph is encoded with a Graph Attention Network (GAT; Velickovic et al. 2018) and the prediction of type and relation fillers for the SPARQL template is modeled as a classification task over graph nodes, given the conversational context and the decoder hidden state. See Chapter 2 (Section 2.6.4 for a brief introduction to graph attention networks.

Learning All modules outlined above are trained in a multi-task manner, optimizing the weighted average of the following individual losses $L = \lambda_1 L^{\mathrm{F}} + \lambda_2 L^{\mathrm{G}} + \lambda_3 L^{\mathrm{R}} + \lambda_4 L^{\mathrm{O}}$ where L^{F} is the loss of the SPARQL template decoder, L^{G} is the type and relation prediction loss using the GAT network, L^{R} is the entity recognition loss, and L^{O} the entity reordering loss (and weights $\lambda_{1:4}$ are learned during training).

4.3 Creating a Knowledge Graph from the CSQA Data

Deploying a full copy of Wikidata locally as a standalone service needs significant resources; Wikimedia's standard process takes about 24 days to complete, split between importing the data and query service updates [7]. To enable easier deployment and fast access for research purposes we created a smaller graph from the CSQA data files. We mapped the contents of these files[8] onto triples which we subsequently converted to `ttl` format[9] with full URI to allow loading them to the KG query engine. We also filled missing information where possible, for example, missing relations such as "instance of" was filled with relation P31 and added data type information when this was omitted from the original files. We used Blazegraph[10] to deploy the local server, which uses only CPU-based resources and has access to 150G of RAM.

4.4 Model Configuration

Both models are implemented using pytorch (Paszke et al., 2019a). For LasagneSP, our model we followed the experimental setup and configuration as Kacupaj et al. (2021)

[6] This graph would be substantially bigger for a semantic parsing system operating over the full Wikidata KG.

[7] `https://www.mediawiki.org/wiki/Wikidata_Query_Service/User_Manual#Standalone_service`

[8] Described at `https://amritasaha1812.github.io/CSQA/download_CQA/`

[9] http://www.w3.org/TR/turtle/

[10] https://blazegraph.com/

	BertSP$_G$		BertSP$_S$		BertSP$_A$		LasagneSP	
Question Type	F1	EM	F1	EM	F1	EM	F1	EM
Clarification	84.89	82.53	80.21	**77.69**	83.91	76.58	86.29	73.41
Logical Reasoning	90.61	82.90	85.55	**66.89**	22.74	28.61	88.80	57.41
Quantitative Reasoning	94.42	88.55	82.95	66.40	76.20	59.01	94.90	**91.47**
Comparative Reasoning	96.23	87.39	90.44	73.80	69.56	39.37	94.20	**85.05**
Simple Question (Coreferenced)	88.96	86.53	83.19	**69.87**	76.51	58.83	84.73	60.90
Simple Question (Direct)	91.81	91.59	87.13	**80.69**	71.43	58.71	87.21	66.88
Simple Question (Ellipsis)	79.51	89.71	72.50	**71.67**	58.14	50.90	74.35	61.53
	AC	EM	AC	EM	AC	EM	AC	EM
Verification (Boolean)	90.10	77.24	79.72	**62.62**	37.16	24.90	34.89	26.72
Quantitative Reasoning (Count)	87.91	84.97	76.88	**73.20**	50.86	48.44	60.51	56.15
Comparative Reasoning (Count)	90.05	85.99	73.18	66.79	43.48	40.67	89.09	**83.69**
Overall	81.50	85.74	81.18	**70.96**	59.00	48.60	79.50	66.32

Table 4.4: Accuracy (AC), and exact match (EM) on SPICE i.i.d test split. BertSP$_G$ has access to oracle entities, types, and coreference annotations. BertSP$_S$ uses string match for entity linking. BertSP$_A$ uses AllenNLP entity recognition and elastic search for entity linking. LasagneSP performs sequence labeling followed by elastic search to link entities. Best EM predictions are shown in bold.

For BertSP experiments, we used the ADAM optimizer (Kingma and Ba, 2015a) with 20,000 BERT warmup steps and 10,000 steps for decoder warm up. We use separate optimizers for the BERT encoder and decoder. BERT was fine-tuned during training with the initial learning rate set to 0.00002. A learning rate of 0.001 was set for the rest of model parameters. Our model was trained for 100,000 steps; we used 4 GPUs with 12GB of memory. *We performed model selection on the validation set.* We report results with the best performing model which had 6 decoder layers.

4.5 Results

We examine how the two models just described fare on different question types and subtypes. We report results on SPICE i.i.d train / valid / test splits (containing 152,391 / 16,813 / 27,797 conversations, respectively) but also create new splits that assess out-of-distribution generalization. In all cases, following previous work (Saha et al., 2018b; Kacupaj et al., 2021), we use execution-based automatic metrics. Micro *F1-score* evaluates question parses that return a set of entities, while *Accuracy* is used for

question parses that evaluate to True/False or return a numerical value. In addition, we report *Exact Match* (EM) against the gold SPARQL parse.

4.5.1 Performance per Question Type

Table 4.4 shows our results on the SPICE i.i.d test split. BertSP$_{\mathrm{G}}$ has access to oracle entities, types, and coreference annotations which allows us to disassociate the complexity of the SPARQL generation task from the problem of grounding and disambiguating entities to KG symbols. Variants BertSP$_{\mathrm{S}}$ and BertSP$_{\mathrm{A}}$ do not have access to oracle annotations. BertSP$_{\mathrm{S}}$ grounds mentions to KG entities with a simple algorithm based on string matching (Marion et al., 2021); while BertSP$_{\mathrm{A}}$ relies on AllenNLP's Named Entity Recognizer (NER) and the Elasticsearch inverted index for Named Entity Linking (NEL). Both have to identify coreferring entities using the conversation context C_t. Both BertSP$_{\mathrm{S}}$ and BertSP$_{\mathrm{A}}$ use string matching for type linking (i.e., grounding general entities to KG symbols). Note that it is not straightforward to perform oracle analysis for LasagneSP without compromising the model structure which predicts entities, their types, and relations in multiple stages. As it requires training a part of the model which is fundamentally a different model compared to proposed LasagneSP's joint optimization.

Exact Match Performance We observe that execution based metrics (F1-score and Accuracy) are generally higher than EM. This is because in some cases the SPARQL parse may be incorrect and still yield some results. For instance, a parse requiring the UNION of two graph patterns may yield a partially correct answer by only including one graph pattern; similarly, a parse can evaluate to False and agree with the gold answer just because it included a wrong relation symbol.

The Importance of Entity Grounding Not surprisingly, the model with access to oracle information (variant BertSP$_{\mathrm{G}}$) obtains the best performance. Results improve not only for questions with entities referring to previous context but also indirectly for other types of questions. Since entities are correctly grounded in previous conversation turns C_t, the model operates with more accurate graphs $\mathcal{G}_t$ and richer dynamic vocabularies $\mathcal{V}_t$.

Both BertSP$_{\mathrm{S}}$ and BertSP$_{\mathrm{A}}$ perform coreference resolution using limited conversation context and thus performance decreases. These models also have to ground named (*Detroit Tigers*) and general (*tournament*) entity mentions to KB symbols. BertSP$_{\mathrm{S}}$ which

Phenomena	BertSP$_{\text{G}}$	BertSP$_{\text{S}}$	BertSP$_{\text{A}}$	LasagneSP
Coreference$_{=-1}$	81.40	70.65	49.39	43.65
Coreference$_{<-1}$	67.82	0	0	0
Ellipsis	75.93	54.33	26.39	46.54
Multiple Entities	83.37	65.40	41.64	66.52

Table 4.5: Exact match on $\mathbb{SPICE}$ i.i.d test; questions grouped into linguistic phenomena. Coreference$_{=-1}$ are the references that can be resolved within the immediate $I[t-1]$, while Coreference$_{<-1}$ are those that needs access to context further in interaction history ($I[t-i]$, where $i > 1$). Ellipsis measure ellipsis resolution performance. Multiple Entities are the utterances that require reasoning over more than one entity.

relies on string matching performs overall better than BertSP$_{\text{A}}$ which struggles with compound named entities such as *President of the Czech Republic*) and disambiguation during NEL (e.g., *Saint Barbara* the painting versus the Saint).

Model Comparison BertSP$_{\text{S}}$ and LasagneSP are similar in the way they handle NER/NEL with a task-specific approach, but differ in their conceptualization of the semantic parsing task (encoder-decoder vs. multi-tasking). LasagneSP outperforms BertSP$_{\text{S}}$ in Comparative, Quantitative, and Comparative-Count questions. These encompass many question sub-types with general entities which are very common in both training and testing. LasagneSP has access to *all* types and relations encoded with the graph network. In contrast, BertSP$_{\text{S}}$ relies on types which in the first place need to be present in the entity neighbourhood subgraph and then be preserved after truncating the input to fit the model's maximum sequence length. An advantage of BertSP$_{\text{S}}$ over LasagneSP, is that it allows for easier adaptation to new types and relations by relying on dynamic vocabularies, while LasagneSP would need to be retrained to accommodate them.

In Simple questions, where each question involves fewer but more diverse types, BertSP$_{\text{S}}$ predicts more accurate types (thanks to the input text and KG symbol contextualisation) and thus performs better. LasagneSP does poorly on Verification, Logical, and Quantitative-Count (which includes logical operators). This can be explained by a modelling limitation, i.e., it is not able to point to the same input entity more than once.

Errors in Predicted SPARQLs Manual inspection of SPARQL predictions revealed several common system errors including: prediction of erroneous entities and relations,

failure to enumerate all required entities (for questions with multiple entities), and mistakes in argument order (i.e., entities and variables are correct but placed in incorrect subject/object positions). To a lesser extent, we also observed SPARQL queries with incorrect intent predictions and ill-formed syntax.

4.5.2 Linguistic Analysis

Table 4.5 shows model performance across-different question sub-types aggregated for specific phenomena. These include coreference, ellipsis, and multiple entities. We distinguish between cases where coreference can be resolved in the previous turn ($I[t-1]$) denoted by Coreference$_{=-1}$ and further back in the conversation history ($I[t-i]$, where $i > 1$) denoted by Coreference$_{<-1}$. In addition, some question sub-types contain plural mentions, i.e., they are linked to multiple entities which the semantic parser must enumerate in order to build the correct parse. Ellipsis can be often resolved within the previous interaction ($I[t-1]$). Questions with multiple entities bring further disambiguation challenges.

As can be seen, the oracle BertSP$_{\text{G}}$ model which has access to gold annotations is superior to variants which rely on automatic entity and type linking. BertSP$_{\text{S}}$ is better than LasagneSP at handling coreference within immediate context (i.e., $I[t-1]$). Due to the fact that LasagneSP predicts entity positions in SPARQL, it is particularly bad at resolving mentions to multiple entities in the previous context or even multiple mentions of the same entity in the output parse (as is the case with Verification questions). Perhaps unsurprisingly, neither BertSP$_{\text{S}}$ nor LasagneSP can resolve mentions to non-immediately preceding utterances. BertSP$_{\text{S}}$ performs better than LasagneSP in questions with ellipsis; we conjecture that the input context and contextualisation of KG symbols help in grounding elided relation mentions. Ellipsis and multiple entities improve by a large margin with access to gold annotations (see BertSP$_{\text{G}}$ in Table 4.5).

4.5.3 Generalisation

We further evaluate the models' ability to generalize by creating "query-based" splits (Finegan-Dollak et al., 2018), i.e., splits with query patterns seen only at test time. Our splits include: (a) question sub-types that involve a count operation over a union operator (COUNTLOGIC; individual operators are seen at training time but not the combination thereof); (b) question sub-types that involve a union operator over two graph patterns with different relations (UNIONMULTI; the union of two graph patterns with the *same* relation

Unseen Combinations	Instances Train/valid/test	BertSP$_S$ EM	LasagneSP EM
CountLogic	153,562/14,262/29,177	0.94	0
UnionMulti	157,331/14,426/25,244	19.74	16.89
Verify3	154,027/13,869/29,105	0	0

Table 4.6: Exact match (EM) for BertSP$_S$ and LasagneSP on SPICE non-i.i.d splits. Each split defines a query pattern with individual reasoning observed during training and combined patterns is seen only at test. See Section 4.5.3 for the definitions of CountLogic, UnionMulti, and Verify3.

is seen during training); and (c) verification questions with three entities (Verify3; questions with 2 entities only are seen during training).

As shown in Table 4.6, both BertSP$_S$ and LasagneSP perform poorly across different splits. While in some cases the models grasp the overall SparQL structure for unseen questions (e.g., *Which watercourses are located in the neighbourhood of Bremen or are the tributaries of Ob? in* UnionMulti), they ignore specific query details and simply default to familiar patterns seen in the training (e.g., *Which people are the creators of The Theory of Everything or Ten Minutes to Live?*). In the UnionMulti split, the models produce an appropriate SparQL template but systematically copy the *same* relation in *both* graph patterns. BertSP$_S$ performs slightly better than LasagneSP; we hypothesize that contextualised KG embeddings occasionally help the model select different relations. We observe a similar trend for CountLogic and Verify3. In

Table 4.7 shows examples from the generalisation splits: similar question sub-types see during training, unseen question sub-type, and error on prediction. The most common error across different splits is that models use similar SparQLs seen during training but fail to adapt them to the details (entities, types, relations, argument positions) in the unseen question sub-type. Other errors encompass using the incorrect SparQL query (incorrect question intent) and incorrect entities and types.

4.6 Related Work

Much previous work on semantic parsing focuses on mapping stand-alone utterances to logical forms. Relatively few datasets have been constructed for *conversational* semantic-parsing (Suhr et al., 2018; Dahl et al., 1994; Yu et al., 2019b,a) partly due

	CountLogic
Seen	*Union — Single Relation* Which people are the creators of The Theory of Everything or Ten Minutes to Live? SELECT ?x WHERE { {wd:Q15079318 wdt:P162 ?x. ?x wdt:P31 wd:Q502895.} UNION {wd:Q7699260 wdt:P162 ?x. ?x wdt:P31 wd:Q502895.} }
Seen	*Count — Single entity type* How many people starred in Captain America: Civil War? SELECT (COUNT(*) AS ?count) WHERE { wd:Q18407657 wdt:P161 ?x. ?x wdt:P31 wd:Q502895.}
UnSeen	*Count — Logical operators* How many national association football teams or national sports teams represent Slovenia? SELECT (COUNT (DISTINCT ?x) AS ?count) WHERE { {?x wdt:P1532 wd:Q215. ?x wdt:P31 wd:Q6979593.} UNION {?x wdt:P1532 wd:Q215. ?x wdt:P31 wd:Q1194951.} } } **Prediction Error** SELECT (COUNT(DISTINCT ?x) AS ?count) WHERE { {?x wdt:P1532 wd:Q215. ?x wdt:P31 wd:Q6979593 .} UNION {?x wdt:P1532 wd:Q215. ?x wdt:P31 wd:Q6979593.} }
	UnionMulti
Seen	*Union — Single Relation* Which people are the creators of The Theory of Everything or Ten Minutes to Live? SELECT ?x WHERE { {wd:Q15079318 wdt:P162 ?x. ?x wdt:P31 wd:Q502895.} UNION {wd:Q7699260 wdt:P162 ?x. ?x wdt:P31 wd:Q502895.} }
Seen	*Count — Mult. entity type* How many people starred in Django Kill or Shatterday? SELECT (COUNT(DISTINCT ?x) AS ?count) WHERE { {wd:Q1261875 wdt:P161 ?x. ?x wdt:P31 wd:Q502895.} UNION {wd:Q7490688 wdt:P161 ?x. ?x wdt:P31 wd:Q502895.}
UnSeen	*Union — Multiple Relation* Which administrative territories are the origin of Les Chics Types or are the native countries of Robert Kuraś? SELECT ?x WHERE {{wd:Q3231475 wdt:P495 ?x. ?x wdt:P31 wd:Q15617994.} UNION {wd:Q9310937 wdt:P27 ?x. ?x wdt:P31 wd:Q15617994.} } **Prediction Error** SELECT ?x WHERE { {wd:Q3231475 wdt:P495 ?x. ?x wdt:P31 wd:Q15617994.} UNION {wd:Q9310937 wdt:P495 ?x. ?x wdt:P31 wd:Q15617994.} }
	Verify3
Seen	*2 entities, both direct* Is Zugspitze located in Germany? ASK {wd:Q3375 wdt:P17 wd:Q183.}
UnSeen	*3 entities, all direct, 2 are query entities* Is Violet Oakley a civilian of United States of America and Scheden? ASK {wd:Q30 wdt:P27 wd:Q1226556. wd:Q557427 wdt:P27 wd:Q1226556.} **Prediction Error** ASK {wd:Q1226556 wdt:P27 wd:Q30. wd:Q557427 wdt:P27 wd:Q557427.}

Table 4.7: Generalisation splits, unseen question sub-types, support question sub-types seen during training, and example common errors on unseen predictions.

to the difficulty of eliciting annotations in an interactive context. As a result, existing benchmarks are either single-domain or small-scale (see the comparison in Table 4.2). For instance, although ATIS (Suhr et al., 2018) exemplifies several challenging long-range discourse phenomena (Jain and Lapata, 2021), it is restricted to a single domain with a simple database schema. SParC (Yu et al., 2019b) and CoSQL (Yu et al., 2019a) present cross-domain challenges in mapping natural language queries onto SQL, but the conversation length is fairly short and the databases relatively small-scale.

Large KGs, like Wikidata (Vrandečić, 2012), are becoming an increasing valuable source of information. Various question-answering datasets have been recently released (Dubey et al. 2019; Talmor and Berant 2018, *inter alia*), but do not cover conversational queries. The CSQA dataset introduced in Saha et al. (2018b) is conversational and covers a wide range of linguistic phenomena (e.g., ellipsis, coreference) but frames the QA task as an information retrieval problem. Follow-on work (Marion et al., 2021; Kacupaj et al., 2021; Shen et al., 2019) has used hand-crafted grammars to automatically obtain semantic annotations which are are not executable with a real KG engine (e.g., Blazegraph), and cumbersome to extend to new question intents.

4.7 Summary

In this chapter we introduce SPICE, a conversational semantic parsing dataset over knowledge graphs. Our dataset contains SPARQL annotations which are executable on a real KG engine and requires handling complex questions, type, relation, and entity linking on a large scale. Moreover, it showcases multiple linguistic phenomena such as coreference and ellipsis. We establish two strong baselines for the semantic parsing task and present detailed analysis stratifying performance by question type and linguistic phenomena. We also study generalization to unseen intents and create multiple dataset splits with different query patterns.

Both models discussed in this work make simplifying assumptions. BertSP variants need to truncate the linearized graphs to reduce computational cost. LasagneSP works with a smaller graph ontology which can easily fit in memory. However, this restricts the model to predicting seen types or relations which is unrealistic. A real-world semantic parser should ideally have access to the full Wikidata. In the next chapter, we will resort to dynamic graphs for modeling *external* context. We will show that a query-dependent dynamic subgraph creation provides a viable representation for semantic parsing tasks over very large knowledge graphs. Furthermore, we will show that the graph structure

itself is crucial and outperforms simply converting the subgraph into a textual string.

Chapter 5

Conversational Question Answering over Wikidata

In Chapter 3, we presented a memory-based method for conversational semantic parsing over long-range interactions. However, we primarily focused on representing the *discourse* context and assumed that the *external* context could be fully encoded within the model. In Chapter 4, we aimed to address this simplification and created SPICE, a conversational question answering dataset based on the Wikidata knowledge graph (KG). The task involves answering user queries by parsing the utterance into a logical form and executing it over a KG engine to extract the answer. This requires reasoning across the entire Wikidata, which, unlike a small relational database schema, cannot be fully encoded within the model. We further introduced two baseline models, BertSP and LasagneSP, and conducted a thorough performance analysis on various question types and linguistic phenomena. While both models demonstrated promising results, they also relied on certain simplifying assumptions. BertSP variants, for instance, needed to truncate the linearized graphs to manage computational costs. LasagneSP, on the other hand, operated with a smaller graph ontology that could fit in memory, but this limited its ability to predict unseen types or relations. In a real-world application, a question answering system should ideally be able to access an up-to-date knowledge graph in real time and encode the relevant knowledge and structure based on the user query.

In this chapter, we remedy this by presenting dynamic context graphs for modeling *external* KG context, which enable real-time access to the KG as well as the encoding of the graph structure. We first revisit the task and provide an example of user-system interaction from the SPICE dataset illustrated in Figure 5.1. Each interaction consists of a series of utterances that form a coherent discourse and are translated to executable

1. Who starred in Mathias Kneissl ?
   SELECT ?x WHERE { wd:Q3298576 wdt:P161 ?x . ?x wdt:P31 wd:Q502895 . }
   Rainer Werner Fassbinder, Volker Schlöndorff, Hanna Schygulla
2. Who was the director of that work of art ?
   SELECT ?x WHERE { wd:Q3298576 wdt:P57 ?x . ?x wdt:P31 wd:Q502895 . }
   Reinhard Hauff
3. Does Dubashi have that person as actor ?
   ASK { wd:Q76025 wdt:P161 wd:Q24807818 . }
   No
4. Which works of art are Rainer Werner Fassbinder or Laura Esquivel a screenwriter of ?
   SELECT ?x WHERE { { ?x wdt:P58 wd:Q44426 . ?x wdt:P31 wd:Q838948 . }
   UNION { ?x wdt:P58 wd:Q230586 . ?x wdt:P31 wd:Q838948 . } }
   The American Soldier, Lili Marleen, Love Is Colder Than Death ...

Q3298576: Mathias Kneissl, Q76025: Reinhard Hauff, Q24807818: Dubashi, Q44426: Rainer Werner Fassbinder Q230586: Laura Esquivel, Q838948: work of art, Q502895: common name, P161: cast member, P31: instance of, P57: director, P58: screenwriter

Figure 5.1: Example interaction from SPICE dataset with utterances, corresponding SPARQL queries, and answers returned after executing the queries on the Wikidata graph engine. The bottom block shows the KG elements (i.e., graph nodes) involved in this interaction.

semantic parses (in this case SPARQL queries). Interpreting each utterance, and mapping it to the correct parse needs to be situated in a particular context as the exchange proceeds. To answer the question in utterance 2, the system needs to recall that *Mathias Kneissl* is still the subject of the conversation, however, the user is no longer interested in who starred in the film but in who directed it. It is also natural for users to omit previously mentioned information (e.g., through ellipsis or coreference), which would have to be resolved to obtain a complete semantic parse.

In addition to challenges arising from processing contextual information, the semantic parsing task itself involves linking entities, types, and predicates to elements in the KG (e.g., *Mathias Kneissl* to Q3298576) whose topology is often complex with a large number of nodes. Moreover, unlike relational databases, the schema of an entity is not static but dynamically instantiated (Gu et al., 2022). For example, the entity type `person` can have hundreds of relations but only a fraction of these will be relevant for a specific

utterance. Therefore, to generate faithful queries, we cannot rely on memorization and should instead make use of local schema instantiation. For more details and examples, refer to Section 2.1.2.2. In general, narrowing down the set of entities and relations is critical to parsing utterances requiring complex reasoning (i.e., where numerical and logical operators apply over sets of entities).

Our BertSP model presented in Chapter 4, handles the aforementioned challenges by adopting various simplifications and shortcuts. For instance, since it is not feasible to encode the entire KG, only a subgraph relevant to the current utterance is extracted and subsequently linearized and treated as a sequence. Entity type information that is not directly accessible via neighboring subgraphs is obtained through a *global* lookup (essentially a reverse index of all types in the KG). This solution is computationally expensive, as the lookup is performed practically for every user utterance, and does not scale well (the index would have to be recreated every time the KG changed).

In this chapter, we propose a modeling approach for conversational semantic parsing that relies on *dynamic context graphs*. We hypothesize that, compared to linearizing the subgraph, **dynamic context graphs offer a computationally efficient representation and effectively capture *external* KG context**. Our key idea is to represent information about an utterance and its context through a dynamically generated subgraph, wherein the number of nodes varies for each utterance. Moreover, rather than treating the subgraph as a sequence, we exploit its underlying structure and encode it with a graph neural network (Scarselli et al., 2009a; Gori et al., 2005b). To improve generalization, we learn *implicit* node embeddings by aggregating information from neighboring nodes whose embeddings are in turned initialized through a pretrained language model. In addition, we introduce context-dependent type linking, based on the entity and its surrounding context which further helps with type disambiguation. Overall our proposed model consists of a graph encoder and a transformer decoder that reasons over both graph embeddings and user utterance to produce the output semantic parse. In the next Chapter 6, we will demonstrate that this model for representing context also transfers to large decoder-only pre-trained models with billions of parameters, such as Mistral-7B-Instruct-v0.2 (Jiang et al., 2023), as well as to tasks beyond semantic parsing.

Experimental evaluation on the SPICE dataset demonstrates that modeling context dynamically is superior to static approaches, improving performance across the board (i.e., for simple and complex questions requiring comparative or quantitative reasoning). Our results further confirm that modeling the structure of context is better at processing discourse information, (i.e., at handling ellipsis and resolving coreference) and longer

interactions with multiple turns.

5.1 Problem Formulation

Given a general purpose knowledge graph, such as Wikidata, our task is to map user utterances to formal executable queries, SPARQL in our case. We further assume an *interactive* setting where users converse with the system in natural language and the system responds while taking into account what has already been said (see Figure 5.1). The system's response is obtained upon *executing* the query against a graph query engine.

Following the problem formulation for conversational semantic parsing from previous chapters. Let G denote the underlying knowledge graph (KG) and I a single interaction. I consists of a sequence of turns where each turn is represented by $\langle X_t, A_t, Y_t \rangle$ denoting an utterance-answer-query triple at time t (see blocks in Figure 5.1). A user utterance X_t is a sequence of tokens $\langle x_1, x_2, \ldots, x_{|X_t|} \rangle$, where $|X_t|$ is the length of the sequence and each $x_i, i \in [1, |X_t|]$ is a natural language token. Query string Y_t is a sequence of tokens $\langle y_1, y_2, \ldots, y_{|Y_t|} \rangle$, where $|Y_t|$ is the length of the sequence and each $y_i, i \in [1, |Y_t|]$ is a either a token from the SPARQL syntax vocabulary (e.g., SELECT, WHERE) or a KG element $\in G$ (e.g., Q3298576). Answer A_t at time t is the result of executing Y_t against G. Given the interaction history $I[:t-1]$ at turn t and current utterance X_t, our goal is to generate Y_t. This involves understanding X_t in the context of $X_{:t-1}$, $A_{:t-1}$, and G, and learning to generate Y_t based on encoded contextual information.

5.2 Model

Our modeling approach combines three components. We first ground named entities in the user utterance to KG entities, and use these linked entities to extract a subgraph that functions as context (Section 5.2.1). The second component is responsible for type linking in the context of current and previously mentioned named entities (Section 5.2.2). And finally, our semantic parser (Section 5.2.3) that consists of a graph neural network and a transformer decoder learns to map user utterances into SPARQL queries.

5.2.1 Entity Grounding and Disambiguation

We are interested in grounding user utterance X_t to graph G. Since encoding the entire KG is not feasible, we extract a subgraph from G which is relevant to the current turn. To achieve this, we first perform named entity recognition with an off-the-shelf NER system, in our experiments we use AllenNLP (Gardner et al., 2018). We perform named entity linking through efficient string matching[1] (Aho and Corasick, 1975). We find string matching to be effective due to the fact that (Saha et al., 2018b) uses exact entity names during automated dataset construction steps.

A string can be ambiguous, i.e., link to multiple entities. For example, *Rainer Werner Fassbinder* can be linked to *filmmaker* (Q44426) and *movie* (Q33561976). To deal with ambiguity and to increase recall, Perez-Beltrachini et al. (2023b) do not ever commit to a single entity but instead include the top-K matching ones; however, this introduces noise and increased computational cost. Instead, we disambiguate entities based on their popularity in the training set (Shen et al., 2015) and compare the two approaches in Section 5.5. Although the popularity-based disambiguation approach has been shown to be effective for web-scale queries (Shen et al., 2015), we note that such an approach generally requires constant statistics updates and consideration beyond the top results to mitigate bias.

Similar to BertSP in Chapter 4, for each identified KG entity e, we extract triples (e, r, o) and (s, r, e), where s and o denote the subject or object of relation r. For instance, entity *Dubashi* would have triple *(Dubashi, country of origin, India)*. Subjects and objects are further mapped to their types in place of actual entities (i.e., (e, r, o_{type}) and (s_{type}, r, e)). In our example, the triple for *Dubashi* then becomes *(Dubashi, country of origin, country)*, where *country* is type Q6256. We denote the set of typed triples as G_t^{ent}.

5.2.2 Context-Dependent Type Linking

Entities in SPARQL queries have types, for example, (SELECT ?x WHERE { wd:Q3298576 wdt:P161 ?x . ?x wdt:P31 wd:Q502895 . }) in Figure 5.1, KG element Q502895 is a placeholder for the type "common name". Type instances are often present in the one-hop entity neighborhood G_t^{ent}, but can also be more hops away. Perez-Beltrachini et al. (2023b) index *all* KG types and perform a *global* lookup which is computationally expensive, and solely applicable to the KG they are working with. Instead, we perform type linking based on the entities mentioned in the *current* context. We expand the grounded entities to

[1] Our implementation is based on pyahocorasick `https://pypi.org/project/pyahocorasick/`.

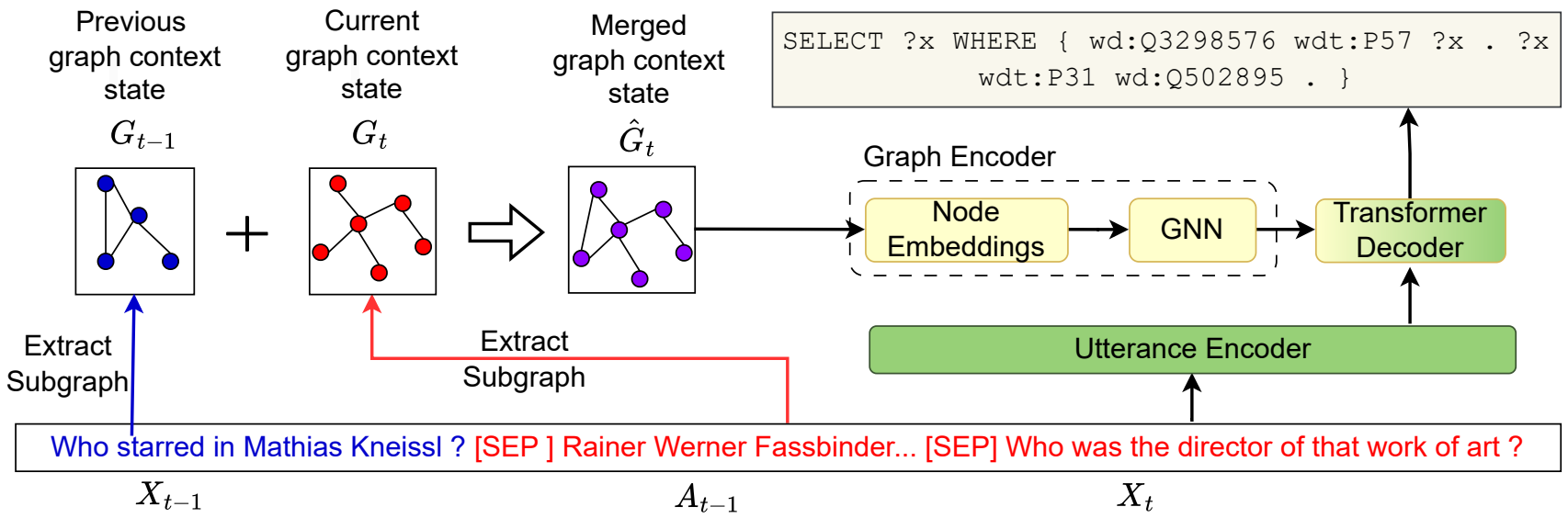

Figure 5.2: Dynamic context graphs based model architecture. The previous and current utterance are concatenated and their subgraphs are merged and encoded in a graph neural network. The subgraphs, denoted by G, represent the entity neighborhood and type linking. G_t is the subgraph at time t that is extracted based on current utterance X_t and previous answer A_{t-1}. Graph G_{t-1} is graph at previous time step $t-1$. Our transformer decoder uses embeddings output from both the utterance and the graph encoder (color gradients are used to visualize this).

extract triples with type information.[2] Since considering multi-hop neighborhoods would lead to extremely large subgraphs and would not be memory efficient, we prune these based on their string overlap with the user utterance, significantly reducing the number of triples. The pruned graph G_t^{type} is merged with the previously obtained entity graph G_t^{ent}, such that, $G_t = G_t^{ent} \cup G_t^{type}$.

5.2.3 Dynamic Context Graph Model

Figure 5.2 shows the overall architecture of our dynamic context graph model (which we abbreviate to DCG). DCG takes as input a tuple of form $\langle C_t, X_t, G_t \rangle$, where X_t is a user utterance at time t and G_t is the corresponding subgraph. C_t denotes the previous context information that includes the previous utterance X_{t-1}, the previous answer A_{t-1}, and subgraph G_{t-1}. We use $\hat{G}_t$ to represent merged context subgraphs G_t and G_{t-1} such that $\hat{G}_t = G_t \cup G_{t-1}$. We encode the context subgraph $\hat{G}_t$ with a graph neural network (GNN, Scarselli et al. 2009a; Gori et al. 2005b) and user utterances and their discourse context with BERT (Devlin et al., 2019). Our decoder is a transformer network (Vaswani et al., 2017a) that conditions on the user utterance encoding and the corresponding graph representations.

Utterance Encoder We use BERT[3] (Devlin et al., 2019) to represent the concatenation of previous utterance X_{t-1}, previous answer A_{t-1}, and current utterance X_t (see Figure 5.2). To distinguish between current and past context we use the [SEP] token. More formally, let $\hat{X}_t$ = [CLS] X_{t-1} [SEP] A_{t-1} [SEP] X_t denote the input to BERT, where $X_t = (x_1, x_2 \ldots x_{|X_t|})$, $A_{t-1} = (a_1, a_2 \ldots a_{|A_{t-1}|})$, and $X_{t-1} = (x_1, x_2 \ldots x_{|X_{t-1}|})$ are sequences of natural language tokens. We obtain latent representations Z_t as $Z_t = \text{BERT}(\hat{X}_t)$.

Graph Encoder We represent the KG subgraph $\hat{G}_t$ at time t as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ (hereafter, we simplify notation and drop time t), where $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ are nodes such that $v_i \in \{entities, relations, types\}$ and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. Each node v_i consists of a sequence of natural language tokens, such that $v_i = \langle v_{i1}, v_{i2}, \ldots, v_{i|v_i|} \rangle$. Our KG has a large number of distinct nodes, but we cannot possibly attest all of

[2] For entity *ent* we query `select ?r1 ?n1 ?t1 ?r2 ?n2 ?t2 where { wd:ent ?r1 ?n1 . ?n1 wdt:P31 ?t1 . OPTIONAL {?n1 ?r2 ?n2 . ?n2 wdt:P31 ?t2} }`.

[3] We employ BERT for a fair comparison with prior work. Nonetheless, our model does not have any inherent restrictions that would prevent the use of other pretrained models.

them during training. To handle unseen nodes at test time, we obtain a generic node representation h_i^0 for node v_i, where $h_i^0 = \text{AVG}(\text{BERT}(v_i))$. In other words, we compute encoding h_i^0 by taking the average of the individual token encodings obtained from BERT. We do not create a separate embedding matrix but directly update the BERT representations during learning, which allows us to scale to a large number of (unseen) nodes.

A graph neural network (GNN; see background Section 2.6.4 for a brief introduction to GNNs) learns node representations by aggregating information from its neighboring nodes. Each GNN layer l, takes as input node representations $\{h_i^{l-1} \mid i \in [1,n]\}$ and edges $\mathcal{E}$. The output of each layer is an updated set of node representations $\{h_i^l \mid i \in [1,n]\}$. We use Graph Attention Network v2 (GATv2, Brody et al. 2022) for updating node representations which replaces the static attention mechanism of GAT (Veličković et al., 2018) with a dynamic and more expressive variant. Let $\mathcal{N}_i = \{v_j \in \mathcal{V} \mid (j,i) \in \mathcal{E}\}$ denote the neighbors of node v_i and ω_{ij} the attention score between node h_i and h_j. We calculate attention as a single-layer feedforward neural network, parametrized by a weight vector ξ and weight matrix W:

$$\omega_{ij} = \frac{\exp\left(\psi\left(h_i^{l-1}, h_j^{l-1}\right)\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\psi\left(h_i^{l-1}, h_k^{l-1}\right)\right)} \tag{5.1}$$

The scoring function ψ is computed as follows:

$$\psi\left(h_i^{l-1}, h_j^{l-1}\right) = \xi^T \, \text{LeakyReLU}\left(W \cdot [h_i^{l-1} \oplus h_j^{l-1}]\right) \tag{5.2}$$

where $\oplus$ is the concatenation operation. Attention coefficients corresponding to each node i are then used to compute a linear combination of the features corresponding to neighboring nodes as:

$$h_i^l = \sigma\left(\sum_{j \in \mathcal{N}_i} \omega_{ij} W h_j^{l-1}\right) \tag{5.3}$$

Decoder Our decoder is a transformer network (Vaswani et al., 2017a). Let $H_t^l = \left(h_{1t}^l, h_{2t}^l, \ldots, h_{nt}^l\right)$ denote the sequence of node representations from the last layer of the graph network (recall t here represents an interaction turn). Our decoder models the probability of generating a SPARQL parse conditioned on the graph and input context representations, i.e., $p(Y_t \mid H_t^l, Z_t)$. Generating the SPARQL parse requires generating syntax symbols (such as SELECT, WHERE) and KG elements (i.e., entities, types, and relations). Given decoder state s_i at the i^{th} generation step, the probability of

Nb. instances	197K
Nb. entities	12.8M
Nb. relations	2738
Nb. types	3064
Average turn length	9.5
Average entities per conversation	7.6
Average types per conversations	6.5
Average neighbourhood per turn	181.4 triples

Table 5.1: Statistics of the SPICE dataset. Repeated from Chapter 4 for quick reference.

generating y_i is calculated as:

$$p(y_i \mid y_{<i}, H_t^l, s_i) = \text{softmax}(\zeta_{\mathcal{G}}(y_i \mid y_{<i}, H_t^l, s_i) \oplus \zeta_{\mathcal{S}}(y_i \mid y_{<i}, s_i)) \tag{5.4}$$

where $\oplus$ is the concatenation operation, $\zeta_{\mathcal{G}}$ and $\zeta_{\mathcal{S}}$ are the unnormalized scores of generating a graph node and syntax symbol, respectively. We calculate $\zeta_{\mathcal{S}} = W_1 s_i$, such that $W_1 \in \mathbb{R}^{|V_s| \times d}$, and $|V_s|$ is the SPARQL syntax vocabulary size. We calculate $\zeta_{\mathcal{G}}$ using node embeddings H_t^l, as $s_{\mathcal{G}} = H_t^l s_i$.

Training Our model is trained end-to-end by optimizing the cross-entropy loss. Given training instance $\langle C_t, X_t, Y_t, G_t \rangle$, where X_t is input utterance, G_t is the context graph, Y_t is a sequence of gold output tokens $\langle y_1, y_2, \ldots, y_{|Y_t|} \rangle$ at time t. C_t represents the context information carried forward from the previous time step. We minimize the token-level cross-entropy as:

$$\mathcal{L}(\hat{y}_i) = -log p(y_i \mid X_t, G_t, C_t) \tag{5.5}$$

where $\hat{y}_i$ denotes the predicted output token at decoder step i. In this work, we train the decoder from scratch, in the next Chapter 6, we will present a method to integrate context graphs into large pretrained decoder-only language models.

5.3 Experimental Setup

Dataset We performed experiments on the SPICE dataset introduced in previous Chapter 4 (Section 4.1). SPICE consists of user-system interactions where natural language questions are paired with SPARQL parses and answers provided by the

system correspond to SPARQL execution results (see Figure 5.1). In Table 5.1 we repeat the summary statistics of the dataset from Chapter 4 for quick reference. As can be seen, it contains a large number of training instances, the conversations are relatively long (the average turn length is 9.5), and the underlying KG is sizeable (12.8M entities). SPICE has simple factual questions but also more complicated ones requiring reasoning over sets of triples; it also exemplifies various discourse-related phenomena such as coreference and ellipsis. Please refer to Section 4.3 of Chapter 4 for examples of the types of questions attested in SPICE dataset.

Evaluation Metrics We report both exact match accuracy and F1 (or accuracy depending on question type). We refer the reader to Chapter 2 (Section 2.5) for details on these metrics. Exact match is the percentage of predicted SPARQL queries that string match with the corresponding gold SPARQL. F1 (or answer accuracy) is calculated between execution results of predicted queries and gold queries. For Verification queries and queries involving Quantitative and Comparative Reasoning, we calculate execution answer accuracy. For other types of questions, F1 scores are calculated by treating the results as a set.

Comparison Models We compare against the BertSP semantic parser presented in Chapter 3. BertSP, as the name suggests, is based on BERT (Devlin et al., 2019), it relies on AllenNLP (Gardner et al., 2018) for named entity recognition, and performs entity linking with an ElasticSearch[4] inverted index. It does not explicitly perform named entity disambiguation; instead, it considers the $K = 5$ best matching entities and their neighborhood graphs as part of the vocabulary, and uses a global lookup for type linking. As the size of the linearized subgraph often exceeds BERT's maximum number of input tokens (which is 512), BertSP adopts a workaround where the graph is chunked into several subsequences, and encoded separately. We use BertSP$_{\mathcal{GL}}$ instead of BertSP to explicitly indicate the use of Global Lookup, with $\mathcal{GL}$ serving as a shorthand.

In addition to our full dynamic context graph model which performs Context-dependent Type Linking (DCG$_{\mathcal{CL}}$), we also build a simpler variant (DCG) which only relies on the entity neighborhood subgraph for type information. Moreover, we create two variants of our model, one which disambiguates entities, and another one which does not.

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

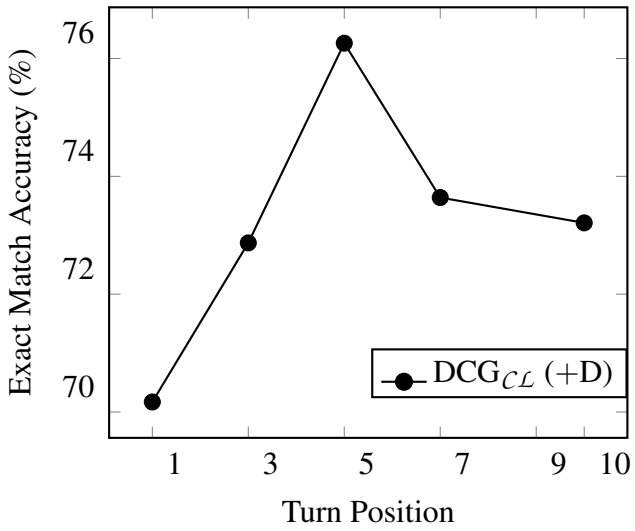

Figure 5.3: Exact Match Accuracy for different context lengths averaged across question types (validation set; +D: with entity disambiguation).

5.4 Model Configuration and Hyperparemeters

Our model is implemented using PyTorch (Paszke et al., 2019a) and trained with the AdamW (Loshchilov and Hutter, 2019) optimizer.[5] Model selection was based on exact match accuracy on the validation set. We used two decoder layers and two GATv2 layers for all experiments. We used HuggingFace's pretrained BERT embeddings (Wolf et al., 2020), specifically the uncased base version. Our GATv2 implementation is based on PyTorch Geometric (Fey and Lenssen, 2019) with two attention heads. We use adjacency matrices stacking as a method of creating mini-batches for our GNN across different examples. We identify named entities using the AllenNLP named entity recognition (NER) system (Gardner et al., 2018). Our execution results are based on the Wikidata subgraph provided by Perez-Beltrachini et al. (2023b).

5.4.1 Hyperparemeters

We use AdamW (Loshchilov and Hutter, 2019) optimizer for learning the weights of our model. Our model is trained on a single A100 GPU with a batch size of 64 and an initial learning rate of 0.001. AdamW coefficients β_1 and β_2 (used for computing running averages of gradient and its square) were set to 0.9 and 0.999, respectively. The weight decay coefficient was set to 0.01 for all experiments. Hyperparameters were set based on initial experiments using a manually selected grid. We did not tune learning rate

[5] Our code can be downloaded from `https://github.com/parajain/dynamic_context`.

parameters. We choose the number of GATv2 and decoder layers from $[1,4]$ and found 2 to work best. Our SPARQL query server was deployed using Blazegraph which uses only CPU-based resources and has access to 100G of RAM.

We use two attention heads with GATv2. Specifically, let K denote the attention head as computed (Veličković et al., 2018) in Equation (5.3). The output of each head is concatenated as follows:

$$h^l = \Big\Vert_{k=1}^{K} \sigma\left(\sum_{j\in\mathcal{N}_i} \omega_{ij}^k W^k h_j^{l-1}\right) \tag{5.6}$$

where $\Vert$ represents concatenation. ω_{ij}^k are normalized attention coefficients computed by the k-th attention mechanism as in Equation (5.1).

Our graph is represented as an adjacency matrix. To create a mini-batch, adjacency matrices are diagonally stacked (Fey and Lenssen, 2019). This creates a combined graph that holds multiple isolated subgraphs as shown below:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 & & \\ & \ddots & \\ & & \mathbf{A}_n \end{bmatrix}$$

where n is the batch-size number of graphs. Node input H and target $\bar{H}$ features are simply concatenated in the node dimension as follows:

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_1 \\ \vdots \\ \mathbf{H}_n \end{bmatrix}, \qquad \bar{\mathbf{H}} = \begin{bmatrix} \bar{\mathbf{H}}_1 \\ \vdots \\ \bar{\mathbf{H}}_n \end{bmatrix}.$$

As described in Section 5.2.3, at each utterance, our model encodes the previous t subgraphs. Larger context is informative but can also introduce noise. We treat t as a hyperparameter and optimize its value on the development set. Figure 5.3 plots the performance of $\text{DCG}_{\mathcal{CL}}$ (disambiguation setting) against progressively increasing context length $\in [1,10]$. We observe that access to wider context is beneficial up to a point. Performance deteriorates with very long contexts (beyond turn position 5). We stipulate two reasons for this. Firstly, longer interactions might be long because users ask about more than one entity or topic, in which case local context might be sufficient to provide an answer. And secondly, longer interactions might be genuinely confusing and noisy for annotators to create, let alone models. Based on this initial result we report results with $t = 5$ in rest of the chapter.

	Without Disambiguation						With Disambiguation			
	$\text{DCG}_{\mathcal{CL}}$		$\text{DCG}_{\mathcal{GL}}$		$\text{BertSP}_{\mathcal{GL}}$		$\text{DCG}_{\mathcal{CL}}$		DCG	
Question Type	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM
Clarification	78.60	73.63	80.42	69.47	**83.91**	**76.58**	82.01	74.82	82.03	72.10
Logical Reasoning	64.12	49.51	51.14	31.54	22.74	28.61	**93.95**	**79.52**	93.33	78.19
Quantitative Reasoning	55.66	26.29	**93.25**	**76.88**	76.20	59.01	59.83	31.17	56.66	28.66
Comparative Reasoning	76.06	35.59	80.59	47.28	69.56	39.37	**90.91**	**62.46**	90.09	61.11
Simple Question (Coref)	86.36	72.03	84.92	67.15	76.51	58.83	**88.49**	**79.90**	87.41	79.18
Simple Question (Direct)	87.29	**71.10**	83.24	65.89	71.43	58.71	**88.27**	62.25	85.60	61.44
Simple Question (Ellipsis)	65.22	56.08	52.84	47.48	58.14	50.90	79.08	**83.87**	**84.35**	82.45
	AC	EM	AC	EM	AC	EM	AC	EM	AC	EM
Verification (Boolean)	78.30	36.97	69.07	21.00	37.16	24.90	**87.41**	**66.32**	86.75	63.66
Quantitative Reasoning (Count)	61.10	56.94	66.59	62.70	50.86	48.44	**75.20**	**70.84**	72.96	69.02
Comparative Reasoning (Count)	42.79	30.40	60.91	44.68	43.48	40.67	**67.70**	**57.34**	66.76	56.60
Overall	69.55	50.85	72.30	53.41	59.00	48.60	**81.28**	**66.85**	80.59	65.24

Table 5.2: Results on $\mathbb{SPICE}$ dataset (test set). $\text{BertSP}_{\mathcal{GL}}$ uses NER based on AllenNLP) and global look-up (subscript $_{\mathcal{GL}}$) for type linking. $\text{DCG}_{\mathcal{CL}}$ uses context-dependent type linking (subscript $_{\mathcal{CL}}$) and also AllenNLP. DCG has no type linking. We measure F1, Accuracy (AC), and Exact Match (EM). Question types are different question categories (see Table 4.3 for examples).

	Context Length 1				Context Length 5				Difference due to type linking			
	$DCG_{\mathcal{CL}}$		DCG		$DCG_{\mathcal{CL}}$		DCG					
Question Type	F1	EM	F1	EM	F1	EM	F1	EM	Δ_{F1_1}	Δ_{EM_1}	Δ_{F1_5}	Δ_{EM_5}
Clarification	75.66	68.61	52.22	53.87	82.01	74.82	82.03	72.1	23.44	14.74	0.02	2.72
Logical Reasoning	92.59	77.07	86.9	67.94	93.95	79.52	93.33	78.19	5.69	9.13	0.62	1.33
Quantitative Reasoning	36.1	13.74	30.91	11.04	59.83	31.17	56.66	28.66	5.19	2.7	3.17	2.51
Comparative Reasoning	76.72	39.75	74.77	37.79	90.91	62.46	90.09	61.11	1.95	1.96	0.82	1.35
Simple Question (Coref)	88.18	79.83	83.04	76.27	88.49	79.9	87.41	79.18	5.14	3.56	1.08	0.72
Simple Question (Direct)	87.56	61.59	79.74	58.21	88.27	62.25	85.6	61.44	7.82	3.38	2.67	0.81
Simple Question (Ellipsis)	80.38	81.75	74.2	76.84	79.08	83.87	84.35	82.45	6.18	4.91	5.27	1.42
	AC	EM	AC	EM	AC	EM	AC	EM	Δ_{AC_1}	Δ_{EM_1}	Δ_{AC_5}	Δ_{EM_5}
Verification (Boolean)	88.02	61.45	82.15	48.24	87.41	66.32	86.75	63.66	5.87	13.21	0.66	2.66
Quantitative Reasoning (Count)	69.41	65.34	65.16	60.58	75.2	70.84	72.96	69.02	4.25	4.76	2.24	1.82
Comparative Reasoning (Count)	42.81	30.5	39.74	28.69	67.7	57.34	66.76	56.6	3.07	1.81	0.94	0.74
Overall	73.74	57.96	66.88	51.95	81.28	66.85	80.59	65.24	6.86	6.01	0.69	1.61

Table 5.3: Interaction of context length and type linking. AC represents Accuracy, and EM is Exact Match. Δ_{F1_1} and Δ_{EM_1} is the absolute difference in F1 and exact match score between $DCG_{\mathcal{CL}}$ and DCG for context length 1. Δ_{F1_5} and Δ_{EM_5} is the absolute F1 and exact match difference for context length 5. We find that the Δ is positive and type linking improves performance.

5.5 Results

In this section, we evaluate the performance of our semantic parser on the SPICE test set. We report results on individual question types and overall. We also analyze our system's ability to handle different discourse phenomena like ellipsis and coreference as well as interactions of varying length.

The Effect of Dynamic Context Table 5.2 summarizes our results. We first concentrate on model variants *without entity disambiguation*. We compare DCG$_{\mathcal{GL}}$, a version of our model which adopts a global lookup for type liking similar to BertSP$_{\mathcal{GL}}$ and differs only in how contextual information is encoded. As we can see, our graph-based model performs better, reaching an F1 score of 72.3% compared to 59% obtained by BertSP$_{\mathcal{GL}}$ which is limited by the way it encodes contextual information. Recall, that BertSP$_{\mathcal{GL}}$ linearizes the subgraph context, splits into subsequences and feeds it to the model chunk-by-chunk. Our model alleviates this problem by efficiently encoding the KG information with a graph neural network, preserving dependencies captured in its structure. As a result, DCG$_{\mathcal{GL}}$ performs better on most question types, including simple questions, and reasoning-style questions. We further compare DCG$_{\mathcal{GL}}$ against a variant which uses context-dependent type linking instead (DCG$_{\mathcal{CL}}$). We find that context-dependent type linking is slightly worse than global lookup which is expected given that it does not have access to the full list of KG types.

In general, we observe that results with exact match (EM) are lower than F1 or Accuracy. EM is a stricter metric, it does not allow for any deviation from the goldstandard SPARQL. However, it is possible for two queries to have different syntax but equivalent meaning, and for partially well-formed queries to evaluate to partially accurate results. In contrast to EM, F1 and Accuracy give partial credit and thus obtain higher scores.

The Effect of Entity Disambiguation We now present results with a variant of our model which operates over *disambiguated entities* (compare second and fifth blocks in Table 5.2, with heading DCG$_{\mathcal{CL}}$). We observe that disambiguation has a significant effect on model performance, leading to an F1 increase of more than 11%. We further assess the utility of context-dependent linking by comparing DCG$_{\mathcal{CL}}$ to a variant which does not have access to the type graph G_t^{type}, neither during training nor during evaluation (see column DCG, With Disambiguation). This type-deficient model performs overall worse both in terms of F1 and EM, but is still superior to BertSP$_{\mathcal{GL}}$, even though the

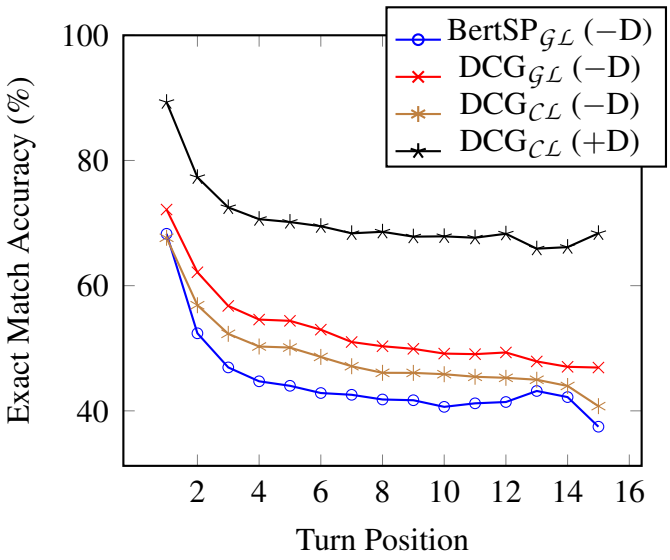

Figure 5.4: Exact Match Accuracy averaged across question types at different turn positions. BertSP$_{\mathcal{GL}}$ chunks the linearized subgraph context to into several subsequences that are encoded separately. DCG is out dynamic graphs based model. Subscript $_{\mathcal{GL}}$ denotes global look-up and subscript $_{\mathcal{CL}}$ denotes context-dependent type linking. $+/-$D denotes the presence/absence of entity disambiguation.

latter has access to more information via the top-K entity lookup and global type linking. This points to the importance of encoding context in a targeted manner rather than brute force.

The Effect of Conversation Length We next examine the benefits of modeling context dynamically. Ideally, a model should produce an accurate semantic parse no matter the conversation length. Figure 5.4 plots exact match accuracy (averaged across question types) against different turn positions. In general, we observe that utterances occurring later in the conversation are more difficult to parse. As the dialogue progresses, subsequent turns become more challenging for the model which is expected to leverage the common ground established so far. This involves maintaining the subgraph context based on the conversation history in addition to handling linguistic phenomena such as coreference and ellipsis. Overall, DCG$_{\mathcal{CL}}$ (with and without entity disambiguation) is superior to BertSP$_{\mathcal{GL}}$, and the gap between the two models is more pronounced for later turns.

We further assess how context length interacts with the availability of type information. Table 5.3 shows the difference in performance with and without explicit type linking at

	Without Disambiguation			With Disambiguation	
	$\text{DCG}_{\mathcal{CL}}$	$\text{DCG}_{\mathcal{GL}}$	$\text{BertSP}_{\mathcal{GL}}$	$\text{DCG}_{\mathcal{CL}}$	DCG
Coref=-1	58.25	51.10	49.39	**74.23**	72.22
Coref< -1	23.52	12.42	00.00	**33.64**	32.40
Ellipsis	39.30	39.90	26.39	**62.26**	61.10
MEntities	43.71	53.46	41.64	**61.59**	58.90

Table 5.4: Average Exact Match accuracy. Coref=-1 are utterances with referring expressions resolved in the previous turn. Coref< -1 are utterances with referring expressions resolved in the wider discourse context, beyond the previous turn and upto the length of the interaction. MEntities abbreviates multiple entities and refers to utterances with plural mentions.

context lengths 1 and 5. As described in Section 5.4, DCG does not have explicit type linking while $\text{DCG}_{\mathcal{CL}}$ uses context-dependent linking, while both models apply entity disambiguation. Δ_{F1_1} is the absolute difference in F1 score between $\text{DCG}_{\mathcal{CL}}$ and DCG for context length 1. Similarly, Δ_{F1_5} denotes the difference for context length 5.

Overall, we find a significant drop in performance for context length 1 compared to context length 5. This indicates that more type information becomes available with increased context length. However, performance varies with question types. Specifically, the exact match difference is lot bigger for Clarification questions compared to Quantitative Reasoning questions which seem to require access to larger KB subgraphs.

Modeling Discourse Phenomena Discourse phenomena, such as ellipsis and coreference, are prevalent in conversations. Ellipsis refers to grammatical omissions from an utterance that can be recovered from context (discussed further in Chapter 2 Section 2.1.1). In the interaction:

> Q1: What does Andrei Neagoe do for a living?
> Q2: And how about Wilhelm Dietrichson?

the phrase *do for a living* is elided from (Q2) but can be understood in the context of (Q1). Coreference on the other hand, occurs between utterances that refer to the same entity. For example, between utterances 2 and 3 in Figure 5.1.

In Table 5.4, using exact match, we assess how different models handle ellipsis and coreference across question types. Coref=-1 refers to cases where coreference can be

resolved in the immediate context, i.e., the previous turn. Coref< -1 involves utterances that require access to wider conversation context, beyond the previous turn and can refer back upto the length of the interaction. In the setting that does not disambiguate entities, we observe that models which exploit discourse context (variants DCG$_{\mathcal{GL}}$ and DCG$_{\mathcal{CL}}$) are better at resolving co-referring and elliptical expressions compared to BertSP$_{\mathcal{GL}}$. We also see that entity disambiguation is very helpful, leading to substantial improvements for DCG$_{\mathcal{CL}}$ across discourse-related phenomena.

We also evaluate model performance on utterances with plural mentions; these are typically linked to multiple entities which the semantic parser must enumerate in order to build a correct parse (MEntities in Table 5.4). DCG$_{\mathcal{CL}}$ with disambiguation is overall best, while DCG (without type linking) is worse. This is not surprising, utterances with multiple entities generally have complex parses, with multiple sub-queries and entity types, which DCG does not have access to.

The Nature of Parsing Errors Overall, we find that our model is able to predict syntactically valid SPARQL queries. Errors are mostly due to misinterpretations of the question's intent given the graph context and previous questions or missing information. Our model also has difficulty parsing Clarification and Quantitative Reasoning questions. For Clarification questions, it is not able to select the right entity after clarification. For example, in the following conversation:

> Answer: Peter G. Van Winkle, Arthur I. Boreman, William E. Stevenson
> Utterance: Which language is that person capable of writing ?
> Clarification: Did you mean Arthur I. Boreman ?
> Answer: No, I meant Peter G. Van Winkle. Could you tell me the answer for that?

it selects *Arthur I. Boreman* (Q709961) instead of *Peter G. Van Winkle* (Q1404201) leading to an incorrect SQL parse. In this case, the broader context overrides useful information in the immediately preceding turn. Determining relevant context based on specific question intents would be helpful, however, we leave this to future work.

Failures in type linking are a major cause of errors for Quantitative Reasoning questions which typically have no or very limited context (e.g., "Which railway stations were managed by exactly 1 social group ?"). However, our model relies on the availability of types in the entity neighborhood, as it performs type linking in a context dependent manner. We observe that it becomes better at parsing such questions when given

access to all KG types (see Table 5.2, $\text{DCG}_{\mathcal{GL}}$ vs. $\text{DCG}_{\mathcal{CL}}$).

5.6 Related Work

Previous work on semantic parsing for KBQA (Gu et al., 2022) has focused on mapping stand-alone utterances to logical form queries. Various approaches have been proposed to this effect which broadly follow three modeling paradigms. Ranking methods first enumerate candidate queries from the KB and then select the query most similar to the utterance as the semantic parse (Ravishankar et al., 2021; Hu et al., 2021; Bhutani et al., 2019). Coarse-to-fine methods (Dong and Lapata, 2018; Ding et al., 2019; Ravishankar et al., 2021) perform semantic parsing in two stages, by first predicting a query sketch, and then filling in missing details. Finally, generation methods (Yin et al., 2021) first rank candidate parses and then predict the final parse by conditioning on the utterance and best retrieved logical forms.

Our task is most related to conversational text-to-SQL parsing, as manifested in datasets like SPARC (Yu et al., 2019b), COSQL (Yu et al., 2019a), and ATIS (Dahl et al., 1994; Suhr et al., 2018). SPARC and COSQL cover multiple domains and include multi-turn user and system interactions. COSQL and SPARC are challenging as they require generalization to unseen databases, but the conversation length is fairly short and the databases relatively small-scale. ATIS contains utterances paired with SQL queries pertaining to a US flight booking task; it exemplifies several long-range discourse phenomena (Jain and Lapata, 2021), however, it covers a single domain with a simple database schema.

Graph-based methods have been previously employed in semantic parsing primarily to encode the database schema, so as to enable the parser to globally reason about the structure of the output query (Bogin et al., 2019). Other work (Hui et al., 2022) uses relational graph networks to jointly represent the database schema and syntactic dependency information from the questions. In the context of conversational semantic parsing, Cai and Wan (2020) use a graph encoder to model how the elements of the database schema interact with information in preceding context. Along similar lines, Hui et al. (2021), use a graph neural network with a dynamic memory decay mechanism to model the interaction of the database schema and the utterances in context as the conversation proceeds. All these approaches encode the schema of relational databases which are significantly smaller in size (e.g., number of entities and types of relations) compared to large-scale KGs, where encoding the entire graph in memory is not feasible.

Closest to our work are methods which cast conversational KGQA as a semantic parsing task (Kacupaj et al., 2021; Marion et al., 2021). These approaches build handcrafted grammars that are not directly executable to a KG engine. Furthermore, they assume the KG can be fully encoded in memory which may not be feasible in real-world settings. In chapter 4, we develop a parser which is executable with a real KG engine (e.g., Blazegraph) but simplify the task by considering only limited conversation context.

In parallel to our work, (Xu et al., 2023) created WikiWebQuestions, a dataset for question answering over Wikidata, with each question is annotated with the corresponding SPARQL query. Their model first attempts to parse the user query into SPARQL using a fine-tuned Llama (Touvron et al., 2023a) model. In case the parsed query does not return any results, they default to GPT-3 (Brown et al., 2020b) for direct answer generation. They show that using a pair of models they can reduce the hallucinations by 10% compared to using only GPT-3. However, their work is restricted to single-turn scenarios. Schneider. et al. (2024) evaluated the performance of different large language models on $\mathbb{SPICE}$ dataset. To avoid computational cost they evaluate on a subset of dataset. Their findings showed that low-rank adaptation (Hu et al., 2022) based parameter efficient fine-tuned LLMs, such as LLaMa-7B (Touvron et al., 2023a), achieve high accuracy with a fraction of training samples. However, they simplified the task by providing the entities and relations required to generated the SPARQL, thus avoiding the large *external* context issue that we are address in this work.

5.7 Summary

In this chapter, we present a semantic parsing approach for question answering over knowledge graphs that interactively maps user utterances into executable logical forms based on prior context. Our model represents information about utterances and their context as KG subgraphs which are created dynamically and encoded using a graph neural network. We further propose a context-dependent approach to type linking which is efficient and scalable. Our experiments reveal that better modeling of contextual information improves performance, in terms of entity and type linking, resolving coreference and ellipsis, and keeping track of the interaction history as the conversation evolves.

Based on the experiments in previous chapters and the results presented here, we have established that *structured representations of context are beneficial for conversational language understanding*. We have explored various methods to encode *discourse* and *external* context in scenarios where external information can and cannot

be fully captured by the model. However, our findings are confined to semantic parsing based methods of question answering. Additionally, our experiments predominantly utilize encoder-decoder models. It remains unclear whether our findings extend to decoder-only models pretrained with billions of parameters, or to tasks beyond semantic parsing. In the next chapter, we will address these concerns and present a method for incorporating structured context representations within pretrained large language models (LLMs).

Chapter 6

Large Language Models and Graph-based Reasoning for Conversational Question Answering

In previous chapters, we investigated various methods for conversational question answering (QA) through semantic parsing, addressing challenges related to different contexts types. We introduced a *discourse* context model in Chapter 3, which used a learned memory module to handle long-range interactions, assuming that external context could be fully encoded. In Chapters 4 and 5, we explored conversational QA over knowledge graphs (KG) and found that query-dependent dynamic subgraphs performed better than linearized representations. These findings highlighted the importance of maintaining explicit context representations for effective conversational language understanding.

However, until now our experiments focused on semantic parsing for conversational question answering (QA). Additionally, we used encoder-decoder based models with approximately 200 million parameters. *It remains unclear whether our findings transfer to decoder-only large language models that are pretrained on web-scale data (Brown et al., 2020b).* In this chapter, we address these concerns and introduce a method for integrating graph-based structured context representations within pretrained large language models (LLMs). We show that our approach outperforms models that lack access to structured information for the task of conversational QA.

Our interest in this thesis is in building systems capable of conversational language understanding, which is fundamental to multiple tasks such as conversational question answering (Chapter 4, 5) or tasks that require taking actions via parsing user utterances

into executable logical forms e.g., flight booking (Chapter 3). Semantic parsing approaches to conversational QA, as discussed previously (Chapters 3, 5), require access to an execution engine, which makes it challenging to apply these modeling solutions across different non-structured modalities. Moreover, the majority of prior work has studied different instantiations of conversational QA, based on the simplifying assumption that answers can be found in a *single* information source. Examples include querying knowledge graphs such as Wikidata (Perez-Beltrachini et al., 2023a; Christmann et al., 2022a; Saha et al., 2018c), identifying answer spans in Wikipedia articles (Reddy et al., 2019; Choi et al., 2018), and searching for answers in table cells (Iyyer et al., 2017b).

In this work, we tackle the key limitations of our previous research by addressing (1) reliance on a single modality and (2) the requirement of access to an execution engine. We first adopting a direct response generation approach to conversational QA (see Chapter 2, Section 2.2.2). Second, unlike previous chapters, where *external* context was limited to a single modality, such as a knowledge graph or database, we extend *external* context to multiple textual modalities, specifically text, tables, infoboxes, and the Wikidata knowledge graph. We focus on the task of conversational question answering over multiple and heterogeneous information sources. Our hypothesis is that **large language models benefit from structured representation of context and that conversational memory is helpful in longer interactions**. Figure 6.1 shows an example interaction from ConvMix (Christmann et al., 2022b), a recently curated dataset, which combines the challenges of understanding questions in context, and retrieving their answers from multiple sources. As can be seen, answers are located in knowledge base triples (response to Q1), infoboxes (responses to Q4 and Q5), and tables (responses to Q2 and Q3). It is also possible for an answer to be found in different sources. Moreover, the interaction in Figure 6.1 displays the hallmarks of naturalistic dialogue. The second question (*Fact Rank?*) can only be interpreted by taking into account the topic of the conversation (i.e., the album *Kid A*) mentioned in the previous utterance. Follow-on questions are short and may seem ungrammatical taken out of context. As the conversation unfolds, the topic shifts from the album *Kid A* to the *Rolling Stone* magazine; Q4 in Figure 6.1 has no dependencies on previous utterances and a hypothetical system would have to recognize that a new topic is being introduced.

Q1: What is the release date of album Kid A?
A1: 2 October 2000

Q2: Fact Rank?
A2: 7

Q3: Ranking on Rolling Stone in 2009?
A3: 1

Q4: Editor?
A4: Noah Shachtman

Q5: Category?
A5: Popular culture

Query at Q3: What is the release date of album Kid A? 2 October 2000 Fact Rank? 7 Ranking on Rolling Stone in 2009?

Example retrieved evidence:

- Rolling Stone, Editor, Noah Shachtman (Infobox)
- Rolling Stone, inception, 1967 (Triple)
- Kid A, publication, 2 October 2000 (Triple)
- Kid A, Publication Rolling Stone, Country US, Accolade The 100 Best Albums of Decade, Year 2009, Rank 1 (Table)
- Kid A, Kid A is the fourth studio album by the English rock band Radio head, released on 2 October 2000 by Parlophone (Text)
- Kid A, Rolling Stone described the Kid A tour as "a revelation, exposing rock and roll humanity" in the songs. (Text)

Figure 6.1: Example interaction (left) from the ConvMix development set (Christmann et al., 2022b) and relevant evidence at query Q3 (right). Utterances Q1–Q3 explore the topic of album *Kid A*. Q4 transitions to the topic of *Rolling Stone* magazine. The evidence is retrieved from diverse sources highlighted in red. Wikipedia text and tables are prepended with their respective article title. Known entities are shown in blue. Underlined entities are identified through string matching.

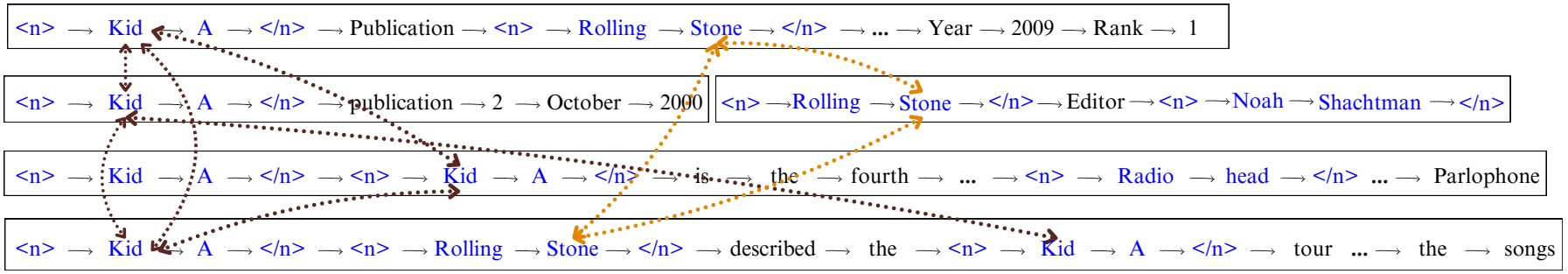

Figure 6.2: Graph for retrieved evidence (subset) from Figure 6.1. Tokens within each instance create local subgraphs in the form of a linear chain. Local subgraphs are connected through common entities (within $\langle n \rangle - \langle /n \rangle$) to build a global graph. Same color highlights connections between similar entities (some edges are omitted for clarity).

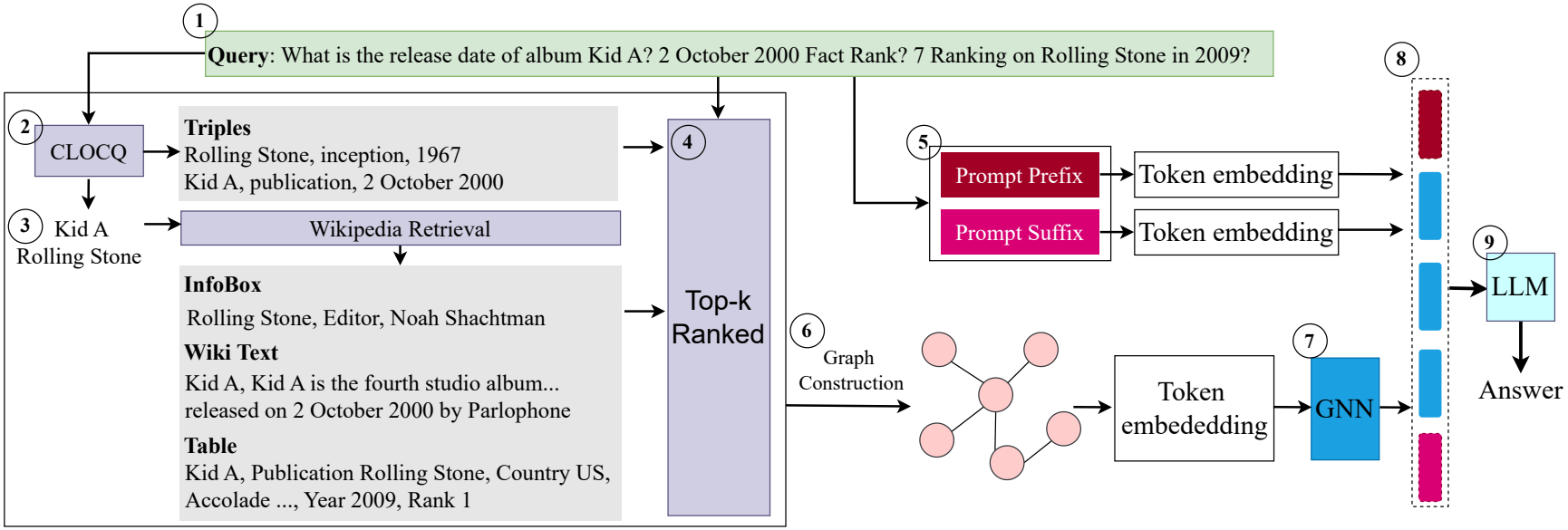

Figure 6.3: Sketch of proposed architecture. ① shows query Q3 from the interaction in Figure 6.1. ② shows KG triples retrieved with CLOCQ and their entities (③). Wikipedia articles for ③ are parsed to extract sentences, infoboxes and tables. In ④, retrieved evidence is ranked based on the current query using BM25. ⑤ creates an instruction prompt based on the input query (see Section 6.3.1 for the prompt template). In ⑥, a graph is constructed based on top ranked instances. ⑦ depicts the learned graph neural network. Graph node embeddings are initialized using LLM token embeddings that are separate from the base model. ⑧ shows the final embeddings which are passed to the LLM and are obtained by concatenating prompt (prefix, suffix) and graph embeddings (shown in different colors). ⑨ is the LLM without the token embedding layer.

We propose a modeling approach to conversational question answering which integrates large language models (LLMs) with graph-based reasoning. Similar to the notion of dynamic context graphs presented in Chapter 5. The core idea is to represent information about a question and its context — such as the conversation so far and sources retrieved to find an answer — through a dynamically generated graph whose size varies with each utterance. Our method utilizes a graph structured representation (Gori et al., 2005c; Scarselli et al., 2009b) to aggregate information (and resolve conflicts) from multiple sources, while also harnessing the reasoning and text generation capabilties of LLMs. Our graph network is efficiently trained using gradients from the LLM. Graph embeddings are directly injected into the LLM, bypassing the token embedding layers, and learned end-to-end by minimizing cross-entropy loss. To manage *topic shifts* (Sun and Chai, 2007a; Jain and Lapata, 2021) and keep track of the conversation flow, we introduce a *memory* module that stores evidence which has been used to answer previous questions, thus allowing to re-use past information for answering future questions. Our contributions are:

- A method to aggregate evidence from multiple sources into a dynamic graph representation for conversational question answering.
- We efficiently integrate the evidence-based graph with LLMs for end-to-end training and
- keep track of past evidence in a memory module which is updated as the conversation evolves and influences the graph structure and its representation.
- Extensive experiments on the ConvMix dataset (Christmann et al., 2022b), demonstrate that graph structure enhances the LLM's ability to reason over multiple sources, while the memory module affords robustness to noise and retrieval errors.

6.1 Problem Formulation

We assume a conversational question answering setting (Christmann et al., 2022b) that requires resoning over Wikipedia facts attested in multiple sources such as text, tables, infoboxes, and the Wikidata knowledge graph (KG). Similar to previous chapters, our discourse context consists of previous user utterances and system response. However, external context consists of facts from both Wikipedia and Wikidata, spanning multiple textual modalities. Based on the user's information need, we retrieve relevant facts from

these sources which are then used to answer the user query.

We follow the problem formulation from Chapter 2 Section 2.2.2. Given interaction I, our task is to answer question X_t at turn t, taking into account retrieved evidence context R_t and previous turns $\mathrm{I}[:t-1]$ which consist of questions and their answers $\langle X_t, A_t \rangle$ (see Figure 6.1). To accommodate information from the conversation so far, we concatenate question q_t at turn t with previous question-answer pairs, i.e., $\hat{X}_t = [X_1, A_1 \ldots X_{t-1}, A_{t-1}, X_t]$, and use this to retrieve evidence. We note that the evidences are retrieved using CLOCQ (Christmann et al., 2022a) rather querying a database or knowledge graph.

As depicted in Figure 6.3, we adopt a modular approach. Given query $\hat{X}_t$, we retrieve and rank relevant evidence (Section 6.2.1). We next organize retrieved information R_t into a graph $\mathcal{G}_\mathrm{t}$ (Section 6.2.3) and learn graph embeddings using Graph Attention Networks (GAT; Veličković et al. 2018; Brody et al. 2022). Finally, graph embeddings are injected in a LLM by skipping the token embeddings layer (Section 6.2.5). Unlike Christmann et al. (2023) who *extract* answers from retrieved evidence, we *generate* them. Our model $\mathcal{M}$ is thus formulated as:

$$A_t = \mathcal{M}\left(\mathrm{I}[:t-1], X_t, \mathcal{G}_\mathrm{t}; \Theta\right) \tag{6.1}$$

where X_t is the current question, $\mathcal{G}_\mathrm{t}$ is the graph representing retrieved evidence, $\mathrm{I}[:t-1]$ are previous turns, and Θ the parameters of our model which are fine-tuned on task-specific data (Section 6.2.6).

6.2 Model

6.2.1 Evidence Retrieval

We adopt the retrieval pipeline outlined in Christmann et al. (2022b). As mentioned earlier, information is obtained from Wikipedia pages and the Wikidata KG using a query based on the current question concatenated with previous question-answer pairs. Retrieval takes place in two stages. Initially, evidence is retrieved from the Wikidata KG, and then followed by retrieval from Wikipedia.

We extract Wikidata triples (see ② in Figure 6.3) using CLOCQ (Christmann et al., 2022a), a retrieval engine specifically tailored to question answering over knowledge bases. It preprocesses the knowledge graph in a memory efficient manner and returns the top-k triples based on query terms. Figure 6.3, shows a subset of relevant triples retrieved for Q3 along with the KG entities $E_\mathcal{E}$.

We next obtain evidence pertaining to additional Wikipedia sources by retrieving articles corresponding to the entities in $E_{\mathcal{E}}$. These pages are subsequently processed to extract text, tables, and infoboxes (see ③ in Figure 6.3). Tables are linearized by individually transforming each row into text and concatenating it with corresponding column headers. Infoboxes are linearized in a similar fashion by concatenating key-value pairs with header information (if available). KB triples are linearized by a simple concatenation of individual elements. Wikipedia text is split into sentences, each of which serves as a separate piece of evidence.

The evidence collected at this stage can be extensive, potentially comprising of several thousand instances, which would in turn lead to a very large graph (see Section 6.2.3). To manage this, we employ BM25 (Robertson and Zaragoza, 2009; Robertson and Jones, 1977) to rank the evidence against the query and retain only the best scoring instances (see ④ in Figure 6.3). Let E_t denote the set of top-k retrieved instances at turn t.

6.2.2 Evidence Memory

By design, we retrieve new evidence at every turn t, which may suggest that every question introduces a new topic. However, a well-known property of conversational dialogue is *topic inertia* (Chai and Jin, 2004), i.e., users tend to explore the same topic for a while before switching to a new topic (see the interaction in Figure 6.1). We propose to keep track of past topics through a memory module which stores previously retrieved pieces of evidence to be re-utilized and re-ranked against $\hat{X}_t$. Specifically, at each turn t we define evidence memory M_t as,

$$M_t = \oplus\{E_j \mid j \in [1 \ldots t-1]\} \tag{6.2}$$

where $\oplus$ denotes concatenation. We replace a portion (e.g., one third) of low-ranked instances from E_t with the top-ranking ones from M_t. We employ the Sentence-BERT model (Reimers and Gurevych, 2019) to re-rank the evidence stored in M_t, using $\hat{X}_t$ as a query. This bounds the computational cost to be proportional to $O(k)$, where k is a hyperparameter used to define the top-k ranked instances discussed in Section 6.2.1.

6.2.3 Graph Construction

Retrieved information is organized into a graph (see ⑥, Figure 6.3) by first converting individual pieces of evidence into a linear chain. *Local* subgraphs are then merged into

a *global* graph by linking common entities between them. Figure 6.2 shows example graphs with *local* and *global* connections.

To construct a *local* graph, evidence from different sources is linearized (as discussed in Section 6.2.1) and tokenized using a base LLM tokenizer. Tokens within each instance are treated as graph nodes connected in a linear chain. In other words, evidence w with tokens $w_1 \ldots w_{|w|}$ is represented by local sub-graph $w_1 \rightarrow w_2 \rightarrow \ldots \rightarrow w_{|w|}$.

Connecting different pieces of evidence together is critical for enabling more global reasoning. We create a *global* graph by linking similar entities across *local* subgraphs. In this context, entities are KG items but also text spans in Wikipedia text, infoboxes, and tables gathered during retrieval. We identify entity spans by performing string matching against KG entities. In Figure 6.2, such entities are encircled by <n> *node* </n> tags. Finally, entity spans referring to same entity are linked, thus creating a more globally connected graph.

6.2.4 Graph Encoder

Our model generates an answer at each turn t given query $\hat{X}_t$ and graph $\mathcal{G}_t$ representing relevant evidence (see Figure 6.3). More formally, $\mathcal{G}_t = (\mathcal{V}, \mathcal{R})$ is a directed graph with nodes $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ and edges $\mathcal{R} \subseteq \mathcal{V} \times \mathcal{V}$.

We do not learn graph node embeddings from scratch. Instead, we initialize them using token embeddings from a large language model (see ⑦, Figure 6.3). This step is crucial for achieving feature alignment between the evidence graph and the downstream LLM. Generally, integrating LLMs with information from a different modality necessitates aligning features between them. For example, vision-language models like BLIP-2 (Li et al., 2023) and LLaVA (Liu et al., 2023) perform feature alignment by heavily pretraining a network whose goal is to act as a bridge between a frozen image encoder and a frozen LLM. This approach requires large amounts of pretraining data (as well as computational resources) which are not readily available for our task. We found that simply initializing graph node embeddings with token embeddings from a base LLM is effective and crucial for achieving good performance.

We adapt the graph neural network formulation introduced in Chapter 2 (Section 2.6.4). Let $\hat{\mathcal{V}} = \{\hat{v}_i \mid i \in [1, n]\}$ denote the set of initial node embeddings for our graph $\mathcal{G}_t$. We learn graph structure representations with the graph attention network (Veličković et al., 2018; Brody et al., 2022)), a neural network architecture designed for

handling graph-structured data.

$$\mathsf{H}_g = \mathrm{GAT}(\hat{\mathcal{V}}, \mathcal{R}) \tag{6.3}$$

where, GAT is the graph attention network that takes as input initial node embeddings $\hat{\mathcal{V}}$ along with edge information $\mathcal{R}$, and returns learned node embeddings H_g for our graph $\mathcal{G}_\mathrm{t}$.

6.2.5 Integration with LLMs

The LLM takes as input a composite embdedding consisting of the graph embeddings discussed above, and embeddings corresponing to a prompt prefix $\mathsf{P}_{\text{prefix}}$, and a prompt suffix $\mathsf{P}_{\text{suffix}}$ (see ⑤ in Figure 6.3). $\mathsf{P}_{\text{prefix}}$ is an initial instruction prompt and $\mathsf{P}_{\text{suffix}}$ represents the conversational query at turn t to be answered. See Section 6.3.1 (Figure 6.5) for an example prompt. More formally, LLM input embeddings are obtained as:

$$\mathsf{H} = \mathsf{H}_{\text{prefix}} \oplus \mathsf{H}_g \oplus \mathsf{H}_{\text{suffix}} \tag{6.4}$$

where H_g is the list of embeddings of all graph nodes and $\mathsf{H}_{\text{prefix}}$ is the text embedding of $\mathsf{P}_{\text{prefix}}$:

$$\mathsf{H}_{\text{prefix}} = \mathrm{Embed}(\mathrm{Tok}(\mathsf{P}_{\text{prefix}})) \tag{6.5}$$

where Tok and Embed are the base LLM tokenizer and embedding layer, respectively. $\mathsf{P}_{\text{suffix}}$ is encoded in a similar manner using Equation (6.5) to obtain $\mathsf{H}_{\text{suffix}}$. We use the embeddings obtained with Equation (6.4) as the initial token embeddings for the pretrained LLM.

6.2.6 Training

Our model is trained end-to-end by optimizing cross-entropy loss. For all variants (with and without graph structure), the loss is calculated on completion tokens only, i.e., prompt tokens do not observe any loss. This is similar to setting the prompt loss weight to 0 (Wang et al., 2023).

Given training instance $\langle \mathsf{I}[:t-1], X_t, \mathcal{G}_\mathrm{t}; \Theta \rangle$, and sequence of gold output tokens $\langle a_t^1, a_t^2, \ldots, a_t^{|A_t|} \rangle$ corresponding to A_t, we minimize token-level cross-entropy as:

$$\mathcal{L}\left(\hat{a}_t^i\right) = -\log p\left(a_t^i \mid \mathsf{I}[:t-1], X_t, \mathcal{G}_\mathrm{t}; \Theta\right) \tag{6.6}$$

where $\hat{a}_t^i$ denotes the predicted output token at decoder step i. We use a mixed approach for training the whole network. Our graph network is trained from scratch, however, the

Parameter	Value
Graph layers	2
Graph heads	2
Lora rank	128
Lora α	32
Lora dropout	0.05
GAT Dropout	0.5
Optimizer	Adam (Kingma and Ba, 2015b)
Learning rate	5e-5
Batch size	1
Gradient accumulation	4

Table 6.1: Hyperparameter values used for our model.

base LLM is updated using LoRA (Hu et al., 2022) in a parameter efficient manner. We perform inference based on the conversation context (i.e., $\mathrm{I}[:t-1]$) and current query X_t.

6.3 Experimental Setup

We use Mistral-7B-Instruct-v0.2 (Jiang et al., 2023) as our base model, given its good performance across complex reasoning tasks, and wider context window of 32K tokens. Recall that we retrieve and encode a large number of instances as evidence for a question. Our implementation predominantly relies on PyTorch (Paszke et al., 2019b). We adapt the Mistral implementation available at the HuggingFace Transformers library (Wolf et al., 2020). For developing the graph neural network, we utilize PyTorch Geometric (PyG; Fey and Lenssen 2019). We use Hugging Face's TRL (Transformer Reinforcement Learning) library (von Werra et al., 2020) for fine-tuning the model without graph. Prompts used in experiments are listed in Section 6.3.1. Table 6.1 lists the hyperparameters employed to train our model. During the fine-tuning of the base language model, only the query, key, and value projection parameters are updated.

ConvMix-5T	
Entities covered	5,418
Long-tail entities	2,511
conversations	2,800
Number of turns	5
Split ratio	60:20:20
ConvMix-10T test set	
Conversations	200
Number of turns	10
Domains: Books, Movies, Music, TV series, Soccer	
Answer Source: Text, Tables, Infobox Wikidata	

Table 6.2: ConvMix dataset statistics. Long tail entities are those attested in less than 50 KG facts.

6.3.1 Prompt Description

Figure 6.4 shows an example prompt for the Mistral-7B model without graph embeddings (see Mistral-7B zero-shot in Table 6.3). The prompt includes a sequence of retrieved and ranked pieces of evidence, each encapsulated within `<evidence>--</evidence>` tags. We represent the past interaction $\mathrm{I}[:t-1]$ as a series of question and answer pairs. The same prompt is used for fine-tuning (see Mistral-7B + FT in Table 6.3) with the subsequent response as the gold output tokens (see Section 6.2.6 for details).

Figure 6.5 shows an example prompt for the graph-based model (all model variants with +Graph in Table 6.3). The prompt consists of three parts, the initial instructions which we refer to as $\mathrm{P}_{\mathrm{prefix}}$, a sequence of graph node embeddings represented as `graph_node_embedding`, and the conversational query which we denote as $\mathrm{P}_{\mathrm{suffix}}$.

6.3.2 Dataset

We evaluate our work on ConvMix (Christmann et al., 2022b), a conversational question answering dataset that requires reasoning over heterogeneous sources, specifically Wikipedia text, infoboxes, tables, and the Wikidata KG. Aside from reasoning, the conversational nature of ConvMix requires handling discourse phenomena, such as coreference, ellipsis, and topic-shift (Sun and Chai, 2007a; Jain and Lapata, 2021). Table 6.2 summarizes various dataset statistics. As can be seen (first block), the main

Prompt: Mistral-7B zero shot and fine-tuned without graph embeddings

[INST]

You are a helpful assistant. Using the following facts:

⟨ evidence⟩Kid A, publication, 2 October 2000 ⟨/evidence⟩

⟨ evidence⟩Rolling Stone, Editor, Noah Shachtman⟨/ /evidence⟩

⟨ evidence⟩Rolling Stone, Catgories, Popular culture ⟨/evidence⟩

⟨ evidence⟩Publication Fact, Country UK, Accolade The 100 Best Albums of the 2000s, Year 2010, Rank 7⟨/evidence⟩

⟨ evidence⟩Publication Rolling Stone, Country US, Accolade The 100 Best Albums of the decade, Year 2009, Rank 1⟨/evidence⟩

⟨ evidence⟩Rolling Stone was founded in San Francisco in 1967 by Jann Wenner and Ralph J. Gleason.⟨/evidence⟩

Answer the following conversational query as a simple key fact without description:

[/INST]

Question: What is the release date of album Kid A?
Answer: 2 October 2000
Question: Fact Rank?
Answer: 7
Question: Ranking on Rolling Stone in 2009?
Answer:

Figure 6.4: Example prompt for models which do not employ graph embeddings. Only a few relevant pieces of evidence are shown, for the sake of brevity.

Prompt: Mistral-7B fine-tuned with graph embeddings

[INST]

You are a helpful assistant. Using the following facts:

`[graph_node_embedding_1, graph_node_embedding_2, ... , graph_node_embedding_n]`

Answer the following conversational query as a simple key fact without description:

[/INST]

Question:	What is the release date of album Kid A?
Answer:	2 October 2000
Question:	Fact Rank?
Answer:	7
Question:	Ranking on Rolling Stone in 2009?
Answer:	

Figure 6.5: Example prompt for graph-based models. We use P_{prefix} and P_{suffix} to denote the instruction before and after the `graph_node_embeddings` respectively. The umber of graph node embeddings is dynamic and varies based on evidence that has been retrieved.

dataset (CovMix-5T) contains 2,800 conversations, each with five turns (i.e., question-answer pairs), split into training, development, and test set. In addition, ConvMix-10T is a *separate* test set used to measure generalization on longer interactions. It contains 200 conversations, each 10 turns long (see last block in Table 6.2). We follow the splits provided in Christmann et al. (2022b) and report results on both test sets combined.

6.3.3 Evaluation Metrics

Our model generates answers which may be valid but not identical to the gold standard (e.g., *United States*, *United States of America*, and *USA* are all paraphrases of the same concept). When there is no exact match, we follow previous work (Christmann et al., 2022b) and try to normalize the answer to its canonical form. We use the Levenshtein distance (Levenshtein, 1965) to measure the similarity of the generated answer with entities in our retrieved evidence set. The entity with the smallest distance is used as the answer in such cases. We report H@1 (i.e., precision at 1) and H@5 (i.e., whether an answer match is found within the top 5 matching entities).

6.4 Results

Our experiments were designed to assess whether graph structure enhances LLM performance for our conversational question-answering task. Our results are summarized in Table 6.3.

We evaluate our approach against Mistral-7B variants without graph structure. Specifically, we compare against (a) Mistral-7B zero-shot prompted with top-k retrieved instances and the conversational history, i.e., the current query concatenated with previous QA pairs (see Section 6.3.1 for the prompt); and (b) Mistral-7B fine-tuned on the ConvMix training set using LoRA (Mistral-7B + FT) and top-k retrieved instances. We present three variants of our model, fine-tined with graph embeddings (Mistral-7B + Graph) and additionally with a memory management component (+Memory, +Rand Memory).

We also compare with several state-of-the-art systems built on top of T5 (Raffel et al., 2020). T5-FiD (Christmann et al., 2022b) is a fusion-in-decoder model which acts as a "generative reader" and is trained on (top-k) retrieved instances and gold answers. Specifically, query-evidence pairs are encoded independently, and passed on to the decoder to generate an answer. We also report results with a T5-based model (T5-FiD + Question rewriting) which rewrites the question based on the conversational history

Models	H@1	H@5
Mistral-7B zero-shot	0.292	0.346
Mistral-7B + FT	0.350	0.400
Mistral-7B + Graph	0.425	0.459
Mistral-7B + Graph + Memory	**0.445**	**0.512**
Mistral-7B + Graph + Rand Memory	0.425	0.461
T5-FiD	0.300	0.350
T5-FiD + Question resolution	0.282	0.297
T5-FiD + Question rewriting	0.271	0.285
Convinse T5-FiD	0.342	0.386
EXPLAIGNN	0.406	0.561

Table 6.3: Model performance on the ConvMix dataset (results are averaged for ConvMix-5T and convMix-10T test sets). H@1 represents precision at 1 and H@5 represents a match at 5. A fine-tuned Mistral-7B with graph embeddings and a memory module performs best.

context (Raposo et al., 2022; Elgohary et al., 2019) and a related approach (T5-FiD + Question resolution) which performs query resolution, i.e., by appending relevant terms from previous question-answer pairs to the current question (Voskarides et al., 2020).[1]

Finally, although not directly comparable, we report the performance of EXPLAIGNN (Christmann et al., 2023) and Convinse T5-FiD (Christmann et al., 2022b). EXPLAIGNN is a classification model that identifies entity nodes in a graph as answer predictions. It learns a task specific structured representation optimized for better retrieval and query understanding. The learned representation is used to train a classification model based on graph neural networks tying both of them together. Convinse T5-FiD is similar in that it also learns a task-specific structured representation for retrieval and query understanding, without, however, creating a graph.

All models in Table 6.3 use the same retrieval engine (i.e., CLOCQ; Christmann et al. 2022a) which allows us to focus on architectural differences and compare models on equal footing.

Integrating LLMs with graph-based reasoning boosts conversational QA perfor-

[1] All FiD models are based on T5-base (Christmann et al., 2022b).

mance. As shown in Table 6.3, Mistral-7B + Graph is superior to a plain fine-tuned version of Mistral-7B (+ FT) by a large margin. This suggests that organizing and representing retrieved evidence as a graph improves reasoning compared to processing pieces of evidence independently. Perhaps unsurprisingly, fine-tuning generally improves Mistral's performance on the conversational QA task over a zero-shot model. This is due to an improved understanding of task requirements, like regular shift in focus and answer format. For example, the model learns to avoid verbosity in answers and respond using dataset-specific conventions such as spelling out the month in dates (e.g., *2 October 2002* instead of *2/10/2002*). The performance of the T5-FiD systems is comparable to zero-shot Mistral-7B. In general, we observe that performance improvements are not simply due to increased model size. Rather, it is important to model the conversational nature of the task and interpret the retrieved information more globally.

Adding a memory module improves QA precision. Table 6.3 shows that results further improve when a memory module is added to our model (+Graph +Memory). Recall that previously retrieved instances are kept in memory and re-reranked against the current query. To further assess the usefulness of re-ranking, we conducted a controlled experiment where evidence was selected randomly from the memory. We observe that random selection (+Rand Memory) amounts to not having a memory component at all.

It is challenging to provide accurate answers to questions that require numerical responses. Figure 6.6(a) shows model performance broken down by question domain. Overall, we observe similar trends across domains, with `TV Series` and `Soccer` being most challenging. Performance for these domains decreases by $\sim$10 percentage points, e.g., in comparison to `Books`. To uncover the reason for this gap, we further investigate whether there is an effect of answer type. We automatically annotate[2] the ConvMix development set with the following answer categories: strings, dates, and numbers. The results in Table 6.4 (top) show average H@1 stratified by different answer types.

We observe that questions with numeric answers are harder compared to other categories. There are several reasons for this, including variability in numerical reasoning performance due to the choice of numeric data tokenization by the base model (Singh and Strouse, 2024; Sun et al., 2023). As well as the effect of pre-training data on the

[2]We use `regex` and `python-dateutil` to automatically categorize the answers.

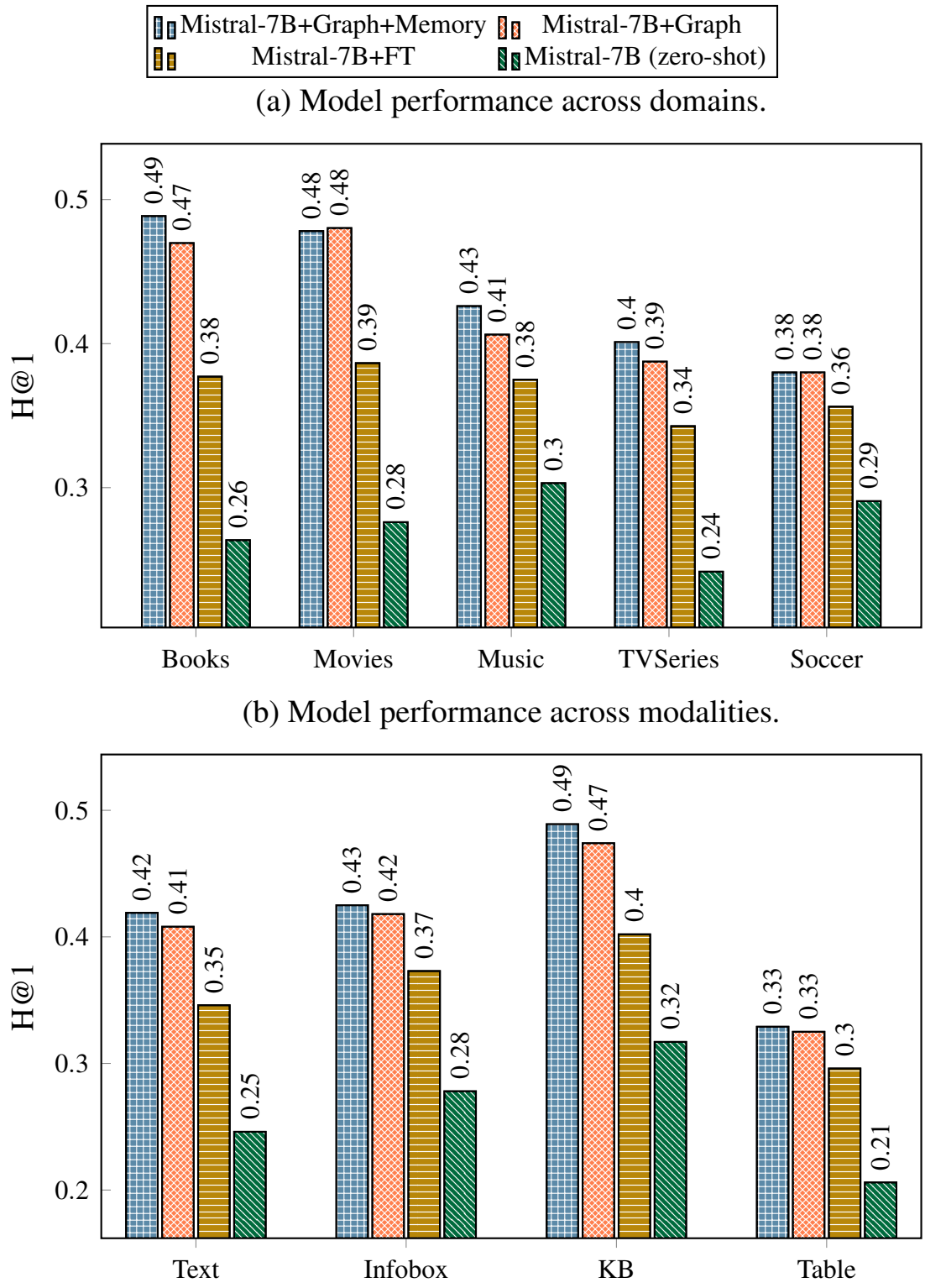

Figure 6.6: Analysis experiments for different model variants based on Mistral-7B prompted in a zero-shot setting, fine-tuned on ConMix without graph embeddings (+FT), with graph embeddings (+Graph), and with a memory module (+Graph +Memory). Low performance in `TV series` and `Soccer` correlates with the percentage of numeric values (see Table 6.4). Tables performs the worst due to challenges in linearization.

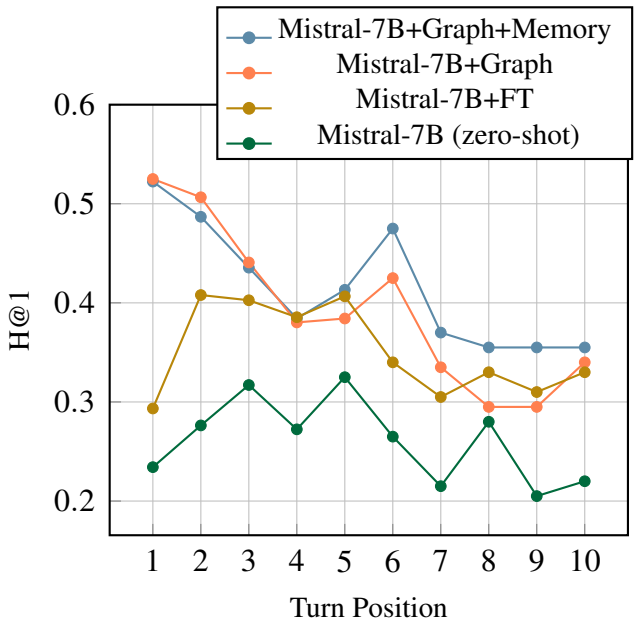

Figure 6.7: Model performance at different turn positions (ConvMix test set). Performance degrades with increase in number of turns across different variants of our model. Conversational memory is helpful for longer turns and zero-shot Mistral-7B performs the worst. We observed a positive shift in performance at turn 6. We believe this improvement is attributable to the extension of the test set, which initially contained dialogues of interaction length 5 (Christmann et al., 2022b).

output predictions and their probability (McCoy et al., 2023). Table 6.4b (bottom) reveals that the proportion of instances with numeric answers is highest for the `TV Series` and `Soccer` domains, thus explaining why performance drops for these domains.

It is challenging to extract accurate information from tables. Figure 6.6(b), shows how performance varies depending on the source of the answer. Across models, we observe that performance deteriorates when the answers are located in tables. On the contrary, performance is generally better when answers are found in the knowledge graph. We believe this performance gap is due to how tabular information is linearized. In contrast to the knowledge graph from which facts can be easily extracted, Wikipedia tables often have complex hierarchical structure (Parikh et al., 2020) making it challenging to achieve clean and robust linearization (Alonso et al., 2023).

It is more difficult to answer questions occurring later in the conversation. In Figure 6.7 we examine how performance varies with conversation length. Ideally, a model

(a)

Answer Type	Date	String	Number
H@1	0.50	0.45	0.14

(b)

Domain	Books	Movies	Music	TV series	Soccer
% Number	3.9	2.1	5.0	10.0	7.9

Table 6.4: Model performance (Mistral-7B + Graph + Memory) across answer types (top) and proportion of numeric answers per domain (ConvMix dev set).

should be able to answer questions irrespective of where these occur (e.g., beginning or end). As mentioned in Section 6.3.2, ConvMix contains conversations with a maximum length of 10 turns. The results in Figure 6.7 show a general decrease in performance as the dialogue progresses. Initial questions tend to be more complex while follow-on questions often extend or elaborate upon the initial topic (Chai and Jin, 2004; Jain and Lapata, 2021). Our results show that graph enhanced models generally outperform LLM variants which do not organize the retrieved information in any way. Furthermore, we observe that having a memory (of previously retrieved instances) is particularly helpful in longer interactions. Keeping track of past evidence helps ameliorate retrieval errors which might erroneously steer the model towards new topics. Aside from contextual factors, the quality of retrieval largely influences model precision, as approximately half of the answers cannot be found even at the beginning of the dialogue (see turn 1 in Figure 6.7).

6.5 Related Work

Conversational Question Answering Most previous work on conversational question answering operates over a *single* infromation source such as a knowledge graph, text passage, or table (Choi et al., 2018; Reddy et al., 2019; Perez-Beltrachini et al., 2023a; Iyyer et al., 2017b). Existing models tend to be specialized, catering to isolated modalities (e.g., text or tables), while a few approaches adopt graph-based representations to organize the conversation and available information (Shen et al., 2019; Jain and Lapata, 2023; Kacupaj et al., 2021; Mueller et al., 2019b). A notable exception are Christmann et al. (2023) who propose an end-to-end model for *multiple* informa-

tion sources. Specifically, their method constructs a heterogeneous graph based on evidence retrieved from tables, infoboxes, text snippets, and Wikidata triples. This graph is iteratively pruned at inference time to a smaller subgraph containing the answer (i.e., an entity node) to the question.

Our work also integrates information from multiple sources into a graph. However, we do not model question answering as a classification task, but instead propose a generative model. We leverage graph representations and the reasoning capabilities of language models, without relying on specialized inference procedures.

LLMs with Graphs A common approach to encoding graph structure for LLMs involves describing the graph in natural language so that it resembles text (Ye et al., 2023; Wang et al., 2024). There is no agreed consensus on how to convert graphs to text, and most methods rely on hand-crafted rules. Previous efforts have shown it is challenging for LLMs to reason over graph representations (Fatemi et al., 2024; Huang et al., 2024), even when explicit prompts are given that describe the structure of the graph in natural language (Huang et al., 2024). Performance tends to be brittle and task dependent (Wang et al., 2024; Fatemi et al., 2024).

Our work proposes a parameter-efficient method for learning *task-specific* graph representations. It is closest to Perozzi et al. (2024), who use graph embeddings as soft-prompts to represent structured data for LLMs. In a similar vein, Chai et al. (2023) use prefix-tuning to integrate graph embeddings with LLM attention layers. Their approach shows promising results on small graphs with a few nodes ($\sim$20) and limited variability. It also relies on the architecture of the LLM and may not seamlessly integrate with other models, e.g., Mixture-of-Experts (MoE; Shazeer et al. 2017; Jacobs et al. 1991).

Retrieval-augmented Generation Our work integrates LLMs with graph structural information based on evidence retrieved from the Wikidata knowledge graph (Vrandečić and Krötzsch, 2014), Wikipedia text, tables, and infoboxes. Although we do not focus on retrieval as such, it plays a key role in identifying information for building the graph. Our approach can thus be viewed as a variant of retrieval augmented generation, since it conditions generation on freshly retrieved evidence based on user queries (Izacard et al., 2024; Khandelwal et al., 2020; Guu et al., 2020).

6.6 Summary

In this chapter, we focus on a conversational question answering task which combines the challenges of understanding questions in context and reasoning over evidence gathered from heterogeneous sources like text, knowledge graphs, tables, and infoboxes. Unlike our work in previous chapters, where we adopted semantic parsing based approach to conversational QA, this work employs direct response generation models. However, as in previous chapters, our task also requires managing long-range dependencies in the form of *discourse* and *external* context. Similar to dynamic context graphs in Chapter 4, we model external context as a query-dependent context graph, extending it to aggregate evidence from multiple sources relevant to the current task. Additionally, we re-use from Chapter 3 the idea of using a context memory to bound the computational cost independently of the interaction length. We introduce a memory module to track and update past evidence. Our memory module further influences the graph's structure and representation, as the conversation evolves.

We demonstrate that our graph based model can be efficiently integrated with large language models (LLMs) for end-to-end training, enhancing the LLM's ability to handle evolving conversational contexts. Experiments on the ConvMix dataset show that the graph enhances the LLM's ability to reason over multiple modalities, while the memory module provides robustness against noise and retrieval errors.

Chapter 7

Conclusion

In this thesis, we are interested in modeling context to improve conversational language understanding of machine learning systems that are learned in an end-to-end manner. This thesis argues that **maintaining explicit representations of context – both external and discourse – is crucial for effective conversational language understanding**, particularly information seeking tasks like question answering (QA). We validate this hypothesis under different scenarios and task constraints, and find that explicit context representations are helpful in a range of problem settings.

In Chapter 3, we tackle the question of explicitly modeling *discourse* context for long-range interactive tasks. We hypothesize that **memory-based representations of discourse are effective in long-range interactive semantic parsing**. We propose a bounded external memory model, which maintains a fixed memory size regardless of the conversation length. Results in ATIS (Suhr et al., 2018), SPARC (Yu et al., 2019b), and COSQL (Yu et al., 2019a) datasets demonstrate our model's ability to perform well across various discourse scenarios, while effectively managing long-range interactions. Apart from basic autoregressive decoders, we also integrate our memory model with task specific decoders, including SnipCopy (Suhr et al., 2018), a model that learns to copy snippets of SQL query generated in previous turns and a grammar-based decoder (Liu et al., 2020a) that uses a SQL grammar for constrained decoding. Our approach consistently improves performance across various decoding architecture and datasets, indicating a better capture of contextual information as an independent language modeling component. The advantages of our memory-based encoder persist across domains and data splits, even with sophisticated strategies like grammar-based decoding. Inspection of model performance of on different discourse phenomena, such as ellipsis, coreference, and focus shift, that are necessary for understanding

extended interactions, shows that our approach is able to handle several discourse related phenomena to a large extent. We further show that our memory is interpretable as we are able to inspect the behavior of the memory controller as the conversation proceeds within an interaction. We observe that memory content and controller decisions correlate with model output. Overall, our study indicates that explicitly modeling *discourse context* can be helpful for contextual language processing tasks.

In Chapter 4, we extend our findings by addressing the integration of *external* context in the form of knowledge graphs. Previous work assumes that *external* context can be fully encoded within a model, which is not feasible for tasks that require interacting with large knowledge graphs, such as Wikidata. In this chapter, we introduce SPICE, a dataset designed for conversational semantic parsing, specifically focusing on the complexities of QA over Wikidata. We propose two baseline models, BertSP and LasagneSP, which tackle the semantic parsing and large vocabulary problem in different ways. BertSP is a sequence-to-sequence model featuring a modified decoder that uses a *dynamic vocabulary* instead of a static one. This approach allows us to predict knowledge graph elements extracted from KG subgraphs for each question. LasagneSP predicts SPARQL *query templates* with placeholder slots for entity, relation, and type. Placeholders are subsequently filled by means of an entity and ontology classifier. Our results on SPICE reveal several shortcomings in both approaches. Both models are unable to encode large sets of KG elements, while LasagneSP is not able to generate the same entity several times in same query.

In Chapter 5, we remedy the problem of encoding large knowledge graphs (Chapter 4) by presenting *dynamic context graphs* for modeling *external* KG context. Dynamic graphs access the KG in real-time, allowing us to encode the latest information without needing any model updates. We represent the utterance and its context through a dynamically generated subgraph, wherein the number of nodes varies for each utterance. We hypothesize that **dynamic context graphs offer a computationally efficient representation and effectively capture *external* KG context**. The underlying structure of the subgraph is captured by encoding it with a graph neural network. Whose node embeddings are initialized from a pretrained language model which allows us to encode unseen entities at test time. We also introduce context-dependent type linking, based on entities and their surrounding context which further helps with type disambiguation. We evaluate our proposed method on the SPICE dataset. Our results show that dynamic context modeling performs better than static approaches. We show superior performance across the board on different question types. Furthermore, our results

confirm that modeling the structure of context is better at processing ellipsis and resolving coreference phenomenon that are required for successfully handling longer interactions with multiple turns.

In Chapter 6, we hypothesize that **large language models benefit from structured representation of context and that conversational memory is helpful in longer interactions**. We demonstrate that, similar to encoder-decoder models in previous works (Cai and Wan (2020); Hui et al. (2021); Jain and Lapata (2023); inter alia), recent decoder-only LLMs also benefit from structured representation of context. We present a method to aggregate evidence from multiple sources into a dynamic graph representation. Our results on the ConvMix dataset (Christmann et al., 2022b) show that graph structure helps LLM's reason over multiple sources. We also propose a memory module to keep track of previous pieces of evidence, which is used to update the context graph allowing it to access relevant information from context history. Overall, our results show that integrating LLMs with graph-based reasoning boosts conversational QA performance over extended interactions. However, we also find that questions that require tabular reasoning are difficult. We believe this is due the challenges in how tabular information is linearized in the presence of complex hierarchical structure (Parikh et al., 2020). Although our proposed memory module is helpful in longer interactions, we find that answering questions occurring later in the conversation remains challenging.

Overall, this thesis provides robust evidence that explicit management of both *external* and *discourse* context significantly improves conversational language understanding. The contributions of this thesis are as follows:

- A memory based model for modeling *discourse context* which allows us to maintain a cumulative meaning of discourse within fixed memory space and process long-range interactions.

- The creation of $\mathbb{SPICE}$, a semantic parsing dataset for conversational semantic parsing over Wikidata. $\mathbb{SPICE}$ is a large-scale dataset with different question types and a wide processing of range of conversational phenomena such as coreference, ellipsis, and topic shift.

- *Dynamic context graphs* as a way of modeling large *external* context. We also propose a scalable method for context-dependent type linking.

- The integration of graph context with large language models for end-to-end training and its interface with a memory module to keep track of previous context.

7.1 Future Work

Data Efficiency Our exploration on conversational question answering methods presented in Chapters 3, 4, and 5, requires training on large datasets. As seen in Chapter 4, annotating such large scale datasets for semantic parsing tasks is costly and time consuming. On the other hand, large language models (LLMs) have shown impressive performance in few-shot learning (Brown et al., 2020b) and efficient fine-tuning (Chen et al., 2024). At the same time, it is evident that current models, such as Mistral-7B-Instruct-v0.2 (Jiang et al., 2023), Gemini-1.0-pro (Team et al., 2023) and Llama-2-70b-chat (Touvron et al., 2023b), cannot handle large conversational context (Maharana et al., 2024). Given these recent developments, we believe further study is needed to fully utilize the potential of the current models in conversational tasks, allowing efficient transfer without large scale task specific annotated datasets. Similar to our work in Chapter 6, integrating a memory module can be effective for conversational tasks beyond question answering.

Structural Information in LLMs In Chapter 6, we find that large pretrained language models, such as Mistral (Jiang et al., 2023), perform better compared to previous approaches (Christmann et al., 2022a) for conversational question answering without task-specific architectural modifications. However, we also observe that these models can be improved by better structural inductive biases. Our proposed model is limited to graph structure and requires fine-tuning on task specific datasets. Recent work has focused on constrained decoding or prompting LLMs to generate various structured modalities such as JSON, Tables, and Graphs (Koo et al., 2024; Jain et al., 2024). However, there is a fundamental gap in LLM capabilities when it comes to reasoning over such modalities. A possible direction is to include structured data, similar to code, while pretraining.

Multi-Agent Systems In Chapter 1, we described an automated travel agent, as being able to perform multiple tasks including, (1) reasoning over different modalities, such as text, tables, and graphs, (2) interacting with external databases and systems via API calls and semantic parsing, and (3) taking actions that changes the world state (for example booking a ticket). Our work addresses the necessary parts separately without ultimately bringing them together in a real-world application. Multiple frameworks have

been proposed for the development of LLM-based multi-agent systems. OpenAgents (Xie et al., 2024b) is an open-source platform to develop language agents for real-world applications. AutoGen (Wu et al., 2023) enables the development of LLM applications using multiple agents that can converse with each other to accomplish certain tasks. However, most success has been seen in constrained simulated environments (Zhou et al., 2023). In the future we would like to have specialized agents that interact with each other within an ecosystem (Ge et al., 2023) to enable a full, end-to-end experience for real-world tasks such as travel planning (Xie et al., 2024a).

Complex Workflows This thesis concentrates on semantic parsing tasks that operate without the need for planning or interaction with multiple systems. However, a fully automated system designed for applications such as travel planning (Xie et al., 2024a) or data wrangling (Yin et al., 2023; Zhang et al., 2024b; You et al., 2025) would demand complex workflows. These workflows frequently require the coordinated use of multiple applications and the manipulation of varied data sources. Enabling such complex workflows require planning and generating code (Jiang et al., 2024) beyond database queries. Recent work in this direction proposes data science notebook-centric LLM (Zhang et al., 2024b) and evaluation benchmarks that require multiple rounds of natural language to code generation in the context of the same *python* notebook. Lei et al. (2025) introduce Spider 2.0, a framework to evaluate real-world text to SQL workflow problems. In the future we would like to extend our work to incorporate such complex, multi-step workflows, exploring how our context management and semantic parsing techniques can be adapted and enhanced to support the generation of intricate code sequences and the management of diverse application interactions.

Bibliography

Abdin, M., Jacobs, S. A., Awan, A. A., Aneja, J., Awadallah, A., Awadalla, H., Bach, N., Bahree, A., Bakhtiari, A., Behl, H., et al. (2024). Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*.

Agrawal, P., Dalmia, A., Jain, P., Bansal, A., Mittal, A., and Sankaranarayanan, K. (2019). Unified semantic parsing with weak supervision. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4801–4810, Florence, Italy. Association for Computational Linguistics.

Aho, A. V. and Corasick, M. J. (1975). Efficient string matching: An aid to bibliographic search. *Commun. ACM*, 18(6):333–340.

Akbik, A., Blythe, D., and Vollgraf, R. (2018). Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Alonso, I., Agirre, E., and Lapata, M. (2023). Pixt3: Pixel-based table to text generation. *arXiv preprint arXiv:2311.09808*.

Androutsopoulos, I., Ritchie, G., and Thanisch, P. (1995). Natural language interfaces to databases – an introduction. *Natural Language Engineering*, 1(1):29–81.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Baddeley, A. D. (2007). *Working memory, thought, and action*. Oxford University Press,, Oxford.

Bahdanau, D., Cho, K., and Bengio, Y. (2015a). Neural machine translation by jointly learning to align and translate. In *ICLR*.

Bahdanau, D., Cho, K., and Bengio, Y. (2015b). Neural machine translation by jointly learning to align and translate. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., Knight, K., Koehn, P., Palmer, M., and Schneider, N. (2013). Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Bengio, Y., Ducharme, R., and Vincent, P. (2000). A neural probabilistic language model. In Leen, T., Dietterich, T., and Tresp, V., editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press.

Bhutani, N., Zheng, X., and Jagadish, H. V. (2019). Learning to answer complex questions over knowledge bases with query composition. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 739–748, New York, NY, USA. Association for Computing Machinery.

Bird, S., Klein, E., and Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc.

Bogin, B., Gardner, M., and Berant, J. (2019). Global reasoning over database structures for text-to-SQL parsing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3659–3664, Hong Kong, China. Association for Computational Linguistics.

Bonatti, P. A., Decker, S., Polleres, A., and Presutti, V. (2019). Knowledge Graphs: New Directions for Knowledge Representation on the Semantic Web (Dagstuhl Seminar 18371). *Dagstuhl Reports*, 8(9):29–111.

Bordes, A., Usunier, N., Chopra, S., and Weston, J. (2015). Large-scale simple question answering with memory networks. *CoRR*, abs/1506.02075.

Brody, S., Alon, U., and Yahav, E. (2022). How attentive are graph attention networks? In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. (1994). Signature Verification using a "Siamese" Time Delay Neural Network. In Cowan, J., Tesauro, G., and Alspector, J., editors, *Advances in Neural Information Processing Systems*, volume 6, pages 737–744. Morgan-Kaufmann.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020a). Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020b). Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Cai, Y. and Wan, X. (2020). IGSQL: Database schema interaction graph based neural model for context-dependent text-to-SQL generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6903–6912, Online. Association for Computational Linguistics.

Castagnola, L. (2002). *Anaphora resolution for question answering*. PhD thesis, Massachusetts Institute of Technology.

Chai, J. Y. and Jin, R. (2004). Discourse structure for context question answering. In *Proceedings of the Workshop on Pragmatics of Question Answering at HLT-NAACL 2004*, pages 23–30, Boston, Massachusetts, USA. Association for Computational Linguistics.

Chai, Z., Zhang, T., Wu, L., Han, K., Hu, X., Huang, X., and Yang, Y. (2023).

GraphIlm: Boosting graph reasoning ability of large language model. *arXiv preprint arXiv:2310.05845*.

Chamberlin, D. D. and Boyce, R. F. (1974). Sequel: A structured english query language. In *Proceedings of the 1974 ACM SIGFIDET (Now SIGMOD) Workshop on Data Description, Access and Control*, SIGFIDET '74, page 249–264, New York, NY, USA. Association for Computing Machinery.

Chen, J., He, J., Chen, F., Lv, Z., Tang, J., Li, W., Liu, Z., Yang, H. H., and Han, G. (2024). Towards general industrial intelligence: A survey on iiot-enhanced continual large models. *arXiv preprint arXiv:2409.01207*.

Chen, J., Yang, L., Raman, K., Bendersky, M., Yeh, J.-J., Zhou, Y., Najork, M., Cai, D., and Emadzadeh, E. (2020a). DiPair: Fast and accurate distillation for trillion-scale text matching and pair modeling. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2925–2937, Online. Association for Computational Linguistics.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. D. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. (2021). Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Chen, Y., Wu, L., and Zaki, M. J. (2020b). Graphflow: Exploiting conversation flow with graph neural networks for conversational machine comprehension. In *IJCAI*.

Cho, K., Courville, A., and Bengio, Y. (2015). Describing multimedia content using attention-based encoder-decoder networks. *IEEE Transactions on Multimedia*, 17(11):1875–1886.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Choi, E., He, H., Iyyer, M., Yatskar, M., Yih, W.-t., Choi, Y., Liang, P., and Zettlemoyer, L. (2018). QuAC: Question answering in context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2174–2184, Brussels, Belgium. Association for Computational Linguistics.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., Garcia, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A. M., Pillai, T. S., Pellat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Diaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K., Eck, D., Dean, J., Petrov, S., and Fiedel, N. (2023). Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Christmann, P., Saha Roy, R., and Weikum, G. (2022a). Beyond ned: Fast and effective search space reduction for complex question answering over knowledge bases. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, WSDM '22. ACM.

Christmann, P., Saha Roy, R., and Weikum, G. (2022b). Conversational question answering on heterogeneous sources. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 144–154, New York, NY, USA. Association for Computing Machinery.

Christmann, P., Saha Roy, R., and Weikum, G. (2023). Explainable conversational question answering over heterogeneous sources via iterative graph neural networks. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, page 643–653, New York, NY, USA. Association for Computing Machinery.

Codd, E. F. (1970). A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387.

Dahl, D. A., Bates, M., Brown, M., Fisher, W., Hunicke-Smith, K., Pallett, D., Pao, C., Rudnicky, A., and Shriberg, E. (1994). Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

Dalton, J., Xiong, C., Kumar, V., and Callan, J. (2020). Cast-19: A dataset for conversational information seeking. In *Proceedings of the 43rd International ACM SIGIR*

Conference on Research and Development in Information Retrieval, SIGIR '20, page 1985–1988, New York, NY, USA. Association for Computing Machinery.

Datta, L. (2020). A survey on activation functions and their relation with xavier and he normal initialization. *arXiv preprint arXiv:2004.06632*.

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

Dey, A. K. (2001). Understanding and using context. *Personal and Ubiquitous Computing*, 5(1):4–7.

Dijk, T. A. v. (2008). *Discourse and Context: A Sociocognitive Approach*. Cambridge University Press.

Ding, J., Hu, W., Xu, Q., and Qu, Y. (2019). Leveraging frequent query substructures to generate formal queries for complex question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2614–2622, Hong Kong, China. Association for Computational Linguistics.

Dong, L. and Lapata, M. (2016). Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43, Berlin, Germany. Association for Computational Linguistics.

Dong, L. and Lapata, M. (2018). Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 731–742, Melbourne, Australia. Association for Computational Linguistics.

Dong, L., Wei, F., Zhou, M., and Xu, K. (2015). Question answering over Freebase with multi-column convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 260–269, Beijing, China. Association for Computational Linguistics.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru, A., Roziere, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong, C., Ferrer, C. C., Nikolaidis, C., Allonsius, D., Song, D., Pintz, D., Livshits, D., Esiobu, D., Choudhary, D., Mahajan, D., Garcia-Olano, D., Perino, D., Hupkes, D., Lakomkin, E., AlBadawy, E., Lobanova, E., Dinan, E., Smith, E. M., Radenovic, F., Zhang, F., Synnaeve, G., Lee, G., Anderson, G. L., Nail, G., Mialon, G., Pang, G., Cucurell, G., Nguyen, H., Korevaar, H., Xu, H., Touvron, H., Zarov, I., Ibarra, I. A., Kloumann, I., Misra, I., Evtimov, I., Copet, J., Lee, J., Geffert, J., Vranes, J., Park, J., Mahadeokar, J., Shah, J., van der Linde, J., Billock, J., Hong, J., Lee, J., Fu, J., Chi, J., Huang, J., Liu, J., Wang, J., Yu, J., Bitton, J., Spisak, J., Park, J., Rocca, J., Johnstun, J., Saxe, J., Jia, J., Alwala, K. V., Upasani, K., Plawiak, K., Li, K., Heafield, K., Stone, K., El-Arini, K., Iyer, K., Malik, K., Chiu, K., Bhalla, K., Rantala-Yeary, L., van der Maaten, L., Chen, L., Tan, L., Jenkins, L., Martin, L., Madaan, L., Malo, L., Blecher, L., Landzaat, L., de Oliveira, L., Muzzi, M., Pasupuleti, M., Singh, M., Paluri, M., Kardas, M., Oldham, M., Rita, M., Pavlova, M., Kambadur, M., Lewis, M., Si, M., Singh, M. K., Hassan, M., Goyal, N., Torabi, N., Bashlykov, N., Bogoychev, N., Chatterji, N., Duchenne, O., Çelebi, O., Alrassy, P., Zhang, P., Li, P., Vasic, P., Weng, P., Bhargava, P., Dubal, P., Krishnan, P., Koura, P. S., Xu, P., He, Q., Dong, Q., Srinivasan, R., Ganapathy, R., Calderer, R., Cabral, R. S., Stojnic, R., Raileanu, R., Girdhar, R., Patel, R., Sauvestre, R., Polidoro, R., Sumbaly, R., Taylor, R., Silva, R., Hou, R., Wang, R., Hosseini, S., Chennabasappa, S., Singh, S., Bell, S., Kim, S. S., Edunov, S., Nie, S., Narang, S., Raparthy, S., Shen, S., Wan, S., Bhosale, S., Zhang, S., Vandenhende, S., Batra, S., Whitman, S., Sootla, S., Collot, S., Gururangan, S., Borodinsky, S., Herman, T., Fowler, T., Sheasha, T., Georgiou, T., Scialom, T., Speckbacher, T., Mihaylov, T., Xiao, T., Karn, U., Goswami, V., Gupta, V., Ramanathan, V., Kerkez, V., Gonguet, V., Do, V., Vogeti, V., Petrovic, V., Chu, W., Xiong, W., Fu, W., Meers, W., Martinet, X., Wang, X., Tan, X. E., Xie, X., Jia, X., Wang, X., Goldschlag, Y., Gaur, Y., Babaei, Y., Wen, Y., Song, Y., Zhang, Y., Li, Y., Mao, Y., Coudert, Z. D., Yan, Z., Chen, Z., Papakipos, Z., Singh, A., Grattafiori, A., Jain, A., Kelsey, A., Shajnfeld, A., Gangidi, A., Victoria, A., Goldstand, A., Menon, A., Sharma, A., Boesenberg, A., Vaughan, A., Baevski, A., Feinstein, A., Kallet, A., Sangani, A., Yunus, A., Lupu, A., Alvarado, A., Caples, A., Gu, A., Ho, A., Poulton, A., Ryan, A., Ramchandani, A., Franco, A., Saraf, A., Chowdhury, A., Gabriel, A., Bharambe, A.,

Eisenman, A., Yazdan, A., James, B., Maurer, B., Leonhardi, B., Huang, B., Loyd, B., Paola, B. D., Paranjape, B., Liu, B., Wu, B., Ni, B., Hancock, B., Wasti, B., Spence, B., Stojkovic, B., Gamido, B., Montalvo, B., Parker, C., Burton, C., Mejia, C., Wang, C., Kim, C., Zhou, C., Hu, C., Chu, C.-H., Cai, C., Tindal, C., Feichtenhofer, C., Civin, D., Beaty, D., Kreymer, D., Li, D., Wyatt, D., Adkins, D., Xu, D., Testuggine, D., David, D., Parikh, D., Liskovich, D., Foss, D., Wang, D., Le, D., Holland, D., Dowling, E., Jamil, E., Montgomery, E., Presani, E., Hahn, E., Wood, E., Brinkman, E., Arcaute, E., Dunbar, E., Smothers, E., Sun, F., Kreuk, F., Tian, F., Ozgenel, F., Caggioni, F., Guzmán, F., Kanayet, F., Seide, F., Florez, G. M., Schwarz, G., Badeer, G., Swee, G., Halpern, G., Thattai, G., Herman, G., Sizov, G., Guangyi, Zhang, Lakshminarayanan, G., Shojanazeri, H., Zou, H., Wang, H., Zha, H., Habeeb, H., Rudolph, H., Suk, H., Aspegren, H., Goldman, H., Damlaj, I., Molybog, I., Tufanov, I., Veliche, I.-E., Gat, I., Weissman, J., Geboski, J., Kohli, J., Asher, J., Gaya, J.-B., Marcus, J., Tang, J., Chan, J., Zhen, J., Reizenstein, J., Teboul, J., Zhong, J., Jin, J., Yang, J., Cummings, J., Carvill, J., Shepard, J., McPhie, J., Torres, J., Ginsburg, J., Wang, J., Wu, K., U, K. H., Saxena, K., Prasad, K., Khandelwal, K., Zand, K., Matosich, K., Veeraraghavan, K., Michelena, K., Li, K., Huang, K., Chawla, K., Lakhotia, K., Huang, K., Chen, L., Garg, L., A, L., Silva, L., Bell, L., Zhang, L., Guo, L., Yu, L., Moshkovich, L., Wehrstedt, L., Khabsa, M., Avalani, M., Bhatt, M., Tsimpoukelli, M., Mankus, M., Hasson, M., Lennie, M., Reso, M., Groshev, M., Naumov, M., Lathi, M., Keneally, M., Seltzer, M. L., Valko, M., Restrepo, M., Patel, M., Vyatskov, M., Samvelyan, M., Clark, M., Macey, M., Wang, M., Hermoso, M. J., Metanat, M., Rastegari, M., Bansal, M., Santhanam, N., Parks, N., White, N., Bawa, N., Singhal, N., Egebo, N., Usunier, N., Laptev, N. P., Dong, N., Zhang, N., Cheng, N., Chernoguz, O., Hart, O., Salpekar, O., Kalinli, O., Kent, P., Parekh, P., Saab, P., Balaji, P., Rittner, P., Bontrager, P., Roux, P., Dollar, P., Zvyagina, P., Ratanchandani, P., Yuvraj, P., Liang, Q., Alao, R., Rodriguez, R., Ayub, R., Murthy, R., Nayani, R., Mitra, R., Li, R., Hogan, R., Battey, R., Wang, R., Maheswari, R., Howes, R., Rinott, R., Bondu, S. J., Datta, S., Chugh, S., Hunt, S., Dhillon, S., Sidorov, S., Pan, S., Verma, S., Yamamoto, S., Ramaswamy, S., Lindsay, S., Lindsay, S., Feng, S., Lin, S., Zha, S. C., Shankar, S., Zhang, S., Zhang, S., Wang, S., Agarwal, S., Sajuyigbe, S., Chintala, S., Max, S., Chen, S., Kehoe, S., Satterfield, S., Govindaprasad, S., Gupta, S., Cho, S., Virk, S., Subramanian, S., Choudhury, S., Goldman, S., Remez, T., Glaser, T., Best, T., Kohler, T., Robinson, T., Li, T., Zhang, T., Matthews, T., Chou, T., Shaked, T., Vontimitta, V., Ajayi, V., Montanez, V., Mohan, V., Kumar, V. S., Mangla, V., Albiero, V., Ionescu, V., Poenaru, V., Mihailescu, V. T., Ivanov,

V., Li, W., Wang, W., Jiang, W., Bouaziz, W., Constable, W., Tang, X., Wang, X., Wu, X., Wang, X., Xia, X., Wu, X., Gao, X., Chen, Y., Hu, Y., Jia, Y., Qi, Y., Li, Y., Zhang, Y., Zhang, Y., Adi, Y., Nam, Y., Yu, Wang, Hao, Y., Qian, Y., He, Y., Rait, Z., DeVito, Z., Rosnbrick, Z., Wen, Z., Yang, Z., and Zhao, Z. (2024). The llama 3 herd of models.

Dubey, M., Banerjee, D., Abdelkawi, A., and Lehmann, J. (2019). Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia. In Ghidini, C., Hartig, O., Maleshkova, M., Svátek, V., Cruz, I., Hogan, A., Song, J., Lefrançois, M., and Gandon, F., editors, *The Semantic Web – ISWC 2019*, pages 69–78, Cham. Springer International Publishing.

Ehrlinger, L. and Wöß, W. (2016). Towards a definition of knowledge graphs. In *International Conference on Semantic Systems*.

Elgohary, A., Peskov, D., and Boyd-Graber, J. (2019). Can you unpack that? learning to rewrite questions-in-context. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5918–5924, Hong Kong, China. Association for Computational Linguistics.

Elhammadi, S., V.S. Lakshmanan, L., Ng, R., Simpson, M., Huai, B., Wang, Z., and Wang, L. (2020). A high precision pipeline for financial knowledge graph construction. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 967–977, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Fang, Y. and Teufel, S. (2016). Improving argument overlap for proposition-based summarisation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 479–485, Berlin, Germany. Association for Computational Linguistics.

Fatemi, B., Halcrow, J., and Perozzi, B. (2024). Talk like a graph: Encoding graphs for large language models. In *The Twelfth International Conference on Learning Representations*.

Fey, M. and Lenssen, J. E. (2019). Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Finegan-Dollak, C., Kummerfeld, J. K., Zhang, L., Ramanathan, K., Sadasivam, S., Zhang, R., and Radev, D. (2018). Improving text-to-SQL evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 351–360, Melbourne, Australia. Association for Computational Linguistics.

Gao, J., Galley, M., and Li, L. (2019a). Neural Approaches to Conversational AI. *Foundations and Trends® in Information Retrieval*, 13(2-3):127–298.

Gao, J., Galley, M., Li, L., et al. (2019b). Neural approaches to conversational AI. *Foundations and Trends® in Information Retrieval*, 13(2-3).

Gardner, M., Grus, J., Neumann, M., Tafjord, O., Dasigi, P., Liu, N. F., Peters, M., Schmitz, M., and Zettlemoyer, L. (2018). AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia. Association for Computational Linguistics.

Ge, Y., Ren, Y., Hua, W., Xu, S., Tan, J., and Zhang, Y. (2023). Llm as os, agents as apps: Envisioning aios, agents and the aios-agent ecosystem. *arXiv e-prints*, pages arXiv–2312.

Gori, M., Monfardini, G., and Scarselli, F. (2005a). A new model for learning in graph domains. In *Proceedings. 2005 IEEE international joint conference on neural networks, 2005.*, volume 2, pages 729–734. IEEE.

Gori, M., Monfardini, G., and Scarselli, F. (2005b). A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734 vol. 2.

Gori, M., Monfardini, G., and Scarselli, F. (2005c). A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734 vol. 2.

Grice, H. P. (1975). *Logic and Conversation*, pages 41 – 58. Brill, Leiden, The Netherlands.

Grosz, B. J., Joshi, A. K., and Weinstein, S. (1995). Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225.

Grosz, B. J. and Sidner, C. L. (1986). Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204.

Gu, Y., Kase, S., Vanni, M., Sadler, B., Liang, P., Yan, X., and Su, Y. (2021). Beyond i.i.d.: Three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*, WWW '21, page 3477–3488, New York, NY, USA. Association for Computing Machinery.

Gu, Y., Pahuja, V., Cheng, G., and Su, Y. (2022). Knowledge base question answering: A semantic parsing perspective.

Guo, D., Tang, D., Duan, N., Zhou, M., and Yin, J. (2018). Dialog-to-action: Conversational question answering over a large-scale knowledge base. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Guo, J., Zhan, Z., Gao, Y., Xiao, Y., Lou, J.-G., Liu, T., and Zhang, D. (2019). Towards complex text-to-SQL in cross-domain database with intermediate representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4524–4535, Florence, Italy. Association for Computational Linguistics.

Guu, K., Lee, K., Tung, Z., Pasupat, P., and Chang, M. (2020). Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

Haller, A., Polleres, A., Dobriy, D., Ferranti, N., and Rodríguez Méndez, S. J. (2022). An analysis of links in wikidata. In Groth, P., Vidal, M.-E., Suchanek, F., Szekley, P., Kapanipathi, P., Pesquita, C., Skaf-Molli, H., and Tamper, M., editors, *The Semantic Web*, pages 21–38, Cham. Springer International Publishing.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hemphill, C. T., Godfrey, J. J., and Doddington, G. R. (1990). The ATIS spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*.

Hobbs, J. (1985). On the coherence and structure of discourse. *CSLI*, 85(37).

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Hoffart, J., Suchanek, F. M., Berberich, K., Lewis-Kelham, E., De Melo, G., and Weikum, G. (2011). Yago2: exploring and querying world knowledge in time, space, context, and many languages. In *Proceedings of the 20th international conference companion on World wide web*, pages 229–232.

Hu, E. J., yelong shen, Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2022). LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Hu, X., Shu, Y., Huang, X., and Qu, Y. (2021). Edg-based question decomposition for complex question answering over knowledge bases. In Hotho, A., Blomqvist, E., Dietze, S., Fokoue, A., Ding, Y., Barnaghi, P., Haller, A., Dragoni, M., and Alani, H., editors, *The Semantic Web – ISWC 2021*, pages 128–145, Cham. Springer International Publishing.

Huang, H.-Y., Choi, E., and Yih, W.-t. (2018). Flowqa: Grasping flow in history for conversational machine comprehension. In *ICLR*.

Huang, J., Zhang, X., Mei, Q., and Ma, J. (2024). Can llms effectively leverage graph structural information through prompts, and why?

Hui, B., Geng, R., Ren, Q., Li, B., Li, Y., Sun, J., Huang, F., Si, L., Zhu, P., and Zhu, X. (2021). Dynamic hybrid relation exploration network for cross-domain context-dependent semantic parsing. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(14):13116–13124.

Hui, B., Geng, R., Wang, L., Qin, B., Li, Y., Li, B., Sun, J., and Li, Y. (2022). S^2SQL: Injecting syntax to question-schema interaction graph encoder for text-to-SQL parsers. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1254–1262, Dublin, Ireland. Association for Computational Linguistics.

Iyer, S., Konstas, I., Cheung, A., Krishnamurthy, J., and Zettlemoyer, L. (2017). Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 963–973, Vancouver, Canada. Association for Computational Linguistics.

Iyyer, M., Yih, W.-t., and Chang, M.-W. (2017a). Search-based neural structured learning for sequential question answering. In *ACL*.

Iyyer, M., Yih, W.-t., and Chang, M.-W. (2017b). Search-based neural structured learning for sequential question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1821–1831, Vancouver, Canada. Association for Computational Linguistics.

Izacard, G., Lewis, P., Lomeli, M., Hosseini, L., Petroni, F., Schick, T., Dwivedi-Yu, J., Joulin, A., Riedel, S., and Grave, E. (2024). Atlas: few-shot learning with retrieval augmented language models. *J. Mach. Learn. Res.*, 24(1).

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.

Jain, P. and Lapata, M. (2021). Memory-Based Semantic Parsing. *Transactions of the Association for Computational Linguistics*, 9:1197–1212.

Jain, P. and Lapata, M. (2023). Conversational semantic parsing using dynamic context graphs. In Bouamor, H., Pino, J., and Bali, K., editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8667–8679, Singapore. Association for Computational Linguistics.

Jain, P. and Lapata, M. (2024). Integrating large language models with graph-based reasoning for conversational question answering. *arXiv preprint arXiv:2407.09506*.

Jain, P., Marzoca, A., and Piccinno, F. (2024). STRUCTSUM generation for faster text comprehension. In Ku, L.-W., Martins, A., and Srikumar, V., editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7876–7896, Bangkok, Thailand. Association for Computational Linguistics.

Jain, S. (2016). Question answering over knowledge base using factual memory networks. In *Proceedings of the NAACL Student Research Workshop*, pages 109–115, San Diego, California. Association for Computational Linguistics.

Jia, R. and Liang, P. (2016). Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22, Berlin, Germany. Association for Computational Linguistics.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., Casas, D. d. l., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., et al. (2023). Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Jiang, J., Wang, F., Shen, J., Kim, S., and Kim, S. (2024). A survey on large language models for code generation. *arXiv preprint arXiv:2406.00515*.

Kacupaj, E., Plepi, J., Singh, K., Thakkar, H., Lehmann, J., and Maleshkova, M. (2021). Conversational question answering over knowledge graphs with transformer and graph attention networks. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 850–862, Online. Association for Computational Linguistics.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. (2020). Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Karlik, B. and Olgac, A. V. (2011). Performance analysis of various activation functions in generalized mlp architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*, 1(4):111–122.

Khandelwal, U., Levy, O., Jurafsky, D., Zettlemoyer, L., and Lewis, M. (2020). Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*.

Kingma, D. P. and Ba, J. (2015a). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kingma, D. P. and Ba, J. (2015b). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kintsch, W. and van Dijk, T. A. (1978). Toward a model of text comprehension and production. *Psychological Review*, 85(5):363–394.

Koo, T., Liu, F., and He, L. (2024). Automata-based constraints for language model decoding. In *First Conference on Language Modeling*.

Krishnan, A. (2018). Making search easier: How amazon's product graph is helping customers find products more easily. *Amazon Blog*, 8.

Lan, Y., He, G., Jiang, J., Jiang, J., Zhao, W. X., and Wen, J.-R. (2021). A survey on complex knowledge base question answering: Methods, challenges and solutions. In Zhou, Z.-H., editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4483–4491. International Joint Conferences on Artificial Intelligence Organization. Survey Track.

Lee, C., Gottschlich, J., and Roth, D. (2021). Toward code generation: A survey and lessons from semantic parsing.

Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., and Bizer, C. (2015). DBpedia – a large-scale, multilingual knowledge base extracted from wikipedia. *Semant. Web*, 6(2):167–195.

Lei, F., Chen, J., Ye, Y., Cao, R., Shin, D., SU, H., SUO, Z., Gao, H., Hu, W., Yin, P., Zhong, V., Xiong, C., Sun, R., Liu, Q., Wang, S., and Yu, T. (2025). Spider 2.0: Evaluating language models on real-world enterprise text-to-SQL workflows. In *The Thirteenth International Conference on Learning Representations*.

Levenshtein, V. I. (1965). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics. Doklady*, 10:707–710.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2020). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Li, J., Li, D., Savarese, S., and Hoi, S. (2023). Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR.

Liang, P. (2013). Lambda dependency-based compositional semantics.

Liang, P., Jordan, M. I., and Klein, D. (2013). Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446.

Lin, K., Bogin, B., Neumann, M., Berant, J., and Gardner, M. (2019). Grammar-based neural text-to-sql generation. *ArXiv*, abs/1905.13326.

Liu, H., Li, C., Wu, Q., and Lee, Y. J. (2023). Visual instruction tuning. In Oh, A., Neumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S., editors, *Advances in Neural Information Processing Systems*, volume 36, pages 34892–34916. Curran Associates, Inc.

Liu, Q., Chen, B., Guo, J., Lou, J.-G., Zhou, B., and Zhang, D. (2020a). How far are we from effective context modeling? an exploratory study on semantic parsing in context. In Bessiere, C., editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3580–3586. International Joint Conferences on Artificial Intelligence Organization. Main track.

Liu, Y. and Lapata, M. (2019). Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3730–3740, Hong Kong, China. Association for Computational Linguistics.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2020b). Ro$\{$bert$\}$a: A robustly optimized $\{$bert$\}$ pretraining approach.

Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Ma, X. and Hovy, E. (2016). End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Maas, A. L., Hannun, A. Y., Ng, A. Y., et al. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Atlanta, GA.

Maharana, A., Lee, D.-H., Tulyakov, S., Bansal, M., Barbieri, F., and Fang, Y. (2024). Evaluating very long-term conversational memory of LLM agents. In Ku, L.-W., Martins, A., and Srikumar, V., editors, *Proceedings of the 62nd Annual Meeting*

of the Association for Computational Linguistics (Volume 1: Long Papers), pages 13851–13870, Bangkok, Thailand. Association for Computational Linguistics.

Mann, W. C. and Thompson, S. A. (1988). Rhetorical structure theory: Toward a functional theory of text organization. *Text-interdisciplinary Journal for the Study of Discourse*, 8(3):243–281.

Marion, P., Nowak, P., and Piccinno, F. (2021). Structured context and high-coverage grammar for conversational question answering over knowledge graphs. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8813–8829, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Mazzone, M. (2015). Constructing the context through goals and schemata: top-down processes in comprehension and beyond. *Frontiers in Psychology*, 6.

McCarthy, J. (1993). Notes on formalizing context. In *Proceedings of the 13th International Joint Conference on Artifical Intelligence - Volume 1*, IJCAI'93, page 555–560, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

McCoy, R. T., Yao, S., Friedman, D., Hardy, M., and Griffiths, T. L. (2023). Embers of autoregression: Understanding large language models through the problem they are trained to solve.

Mikolov, T. and Zweig, G. (2012). Context dependent recurrent neural network language model. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 234–239. IEEE.

Miller, S., Stallard, D., Bobrow, R., and Schwartz, R. (1996). A fully statistical approach to natural language interfaces. In *34th Annual Meeting of the Association for Computational Linguistics*, pages 55–61, Santa Cruz, California, USA. Association for Computational Linguistics.

Mitkov, R. (1999). *Anaphora resolution: the state of the art*. School of Languages and European Studies, University of Wolverhampton ....

Mueller, T., Piccinno, F., Shaw, P., Nicosia, M., and Altun, Y. (2019a). Answering conversational questions on structured data without logical forms. In *EMNLP-IJCNLP*.

Mueller, T., Piccinno, F., Shaw, P., Nicosia, M., and Altun, Y. (2019b). Answering conversational questions on structured data without logical forms. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5902–5910, Hong Kong, China. Association for Computational Linguistics.

Neal, R. M. (1992). Connectionist learning of belief networks. *Artificial intelligence*, 56(1):71–113.

Noy, N., Gao, Y., Jain, A., Narayanan, A., Patterson, A., and Taylor, J. (2019). Industry-scale knowledge graphs: lessons and challenges. *Commun. ACM*, 62(8):36–43.

Owens, M. (2006). *The definitive guide to SQLite*. Springer.

Őzcan, F., Quamar, A., Sen, J., Lei, C., and Efthymiou, V. (2020). State of the art and open challenges in natural language interfaces to data. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD '20, page 2629–2636, New York, NY, USA. Association for Computing Machinery.

Parikh, A., Wang, X., Gehrmann, S., Faruqui, M., Dhingra, B., Yang, D., and Das, D. (2020). ToTTo: A controlled table-to-text generation dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186, Online. Association for Computational Linguistics.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019a). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32, pages 8026–8037. Curran Associates, Inc.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019b). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Pennington, J., Socher, R., and Manning, C. (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in*

Natural Language Processing (EMNLP), pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Perez-Beltrachini, L., Jain, P., Monti, E., and Lapata, M. (2023a). Semantic parsing for conversational question answering over knowledge graphs. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2507–2522, Dubrovnik, Croatia. Association for Computational Linguistics.

Perez-Beltrachini, L., Jain, P., Monti, E., and Lapata, M. (2023b). Semantic parsing for conversational question answering over knowledge graphs. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2499–2514, Dubrovnik, Croatia. Association for Computational Linguistics.

Perozzi, B., Fatemi, B., Zelle, D., Tsitsulin, A., Kazemi, M., Al-Rfou, R., and Halcrow, J. (2024). Let your graph do the talking: Encoding structured data for llms. *arXiv preprint arXiv:2402.05862*.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Pourreza, M. and Rafiei, D. (2023). Din-sql: Decomposed in-context learning of text-to-sql with self-correction. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S., editors, *Advances in Neural Information Processing Systems*, volume 36, pages 36339–36348. Curran Associates, Inc.

Qiu, M., Huang, X., Chen, C., Ji, F., Qu, C., Wei, W., Huang, J., and Zhang, Y. (2021). Reinforced history backtracking for conversational question answering. In *AAAI*, volume 35.

Qu, C., Yang, L., Qiu, M., Croft, W. B., Zhang, Y., and Iyyer, M. (2019a). Bert with history answer embedding for conversational question answering. In *SIGIR*.

Qu, C., Yang, L., Qiu, M., Zhang, Y., Chen, C., Croft, W. B., and Iyyer, M. (2019b). Attentive history selection for conversational question answering. In *CIKM*.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Rajkumar, N., Li, R., and Bahdanau, D. (2022). Evaluating the text-to-sql capabilities of large language models. *arXiv preprint arXiv:2204.00498*.

Raposo, G., Ribeiro, R., Martins, B., and Coheur, L. (2022). Question rewriting? Assessing its importance for conversational question answering. In *ECIR*.

Ravishankar, S., Thai, J., Abdelaziz, I., Mihindukulasooriya, N., Naseem, T., Kapanipathi, P., Rossillo, G., and Fokoue, A. (2021). A two-stage approach towards generalization in knowledge base question answering. *CoRR*, abs/2111.05825.

Reddy, S., Chen, D., and Manning, C. D. (2019). CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Robertson, S. and Zaragoza, H. (2009). The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Robertson, S. E. and Jones, K. S. (1977). Relevance weighting of search terms. *Journal of the American Society for Information Science*.

Saha, A., Pahuja, V., Khapra, M., Sankaranarayanan, K., and Chandar, S. (2018a). Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph. In *AAAI*.

Saha, A., Pahuja, V., Khapra, M., Sankaranarayanan, K., and Chandar, S. (2018b). Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph. In McIlraith, S. and Weinberger, K., editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, pages 705–713, New Orleans, Louisiana, USA. AAAI Press.

Saha, A., Pahuja, V., Khapra, M. M., Sankaranarayanan, K., and Chandar, S. (2018c). Complex sequential question answering: towards learning to converse over linked question answer pairs with a knowledge graph. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press.

Sanh, V., Webson, A., Raffel, C., Bach, S., Sutawika, L., Alyafeai, Z., Chaffin, A., Stiegler, A., Raja, A., Dey, M., Bari, M. S., Xu, C., Thakker, U., Sharma, S. S., Szczechla, E., Kim, T., Chhablani, G., Nayak, N., Datta, D., Chang, J., Jiang, M. T.-J., Wang, H., Manica, M., Shen, S., Yong, Z. X., Pandey, H., Bawden, R., Wang, T., Neeraj, T., Rozen, J., Sharma, A., Santilli, A., Fevry, T., Fries, J. A., Teehan, R., Scao, T. L., Biderman, S., Gao, L., Wolf, T., and Rush, A. M. (2022). Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*.

Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T. (2016). Meta-learning with memory-augmented neural networks. In Balcan, M. F. and Weinberger, K. Q., editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1842–1850, New York, New York, USA. PMLR.

Sasaki, Y. et al. (2007). The truth of the f-measure. *Teach tutor mater*, 1(5):1–5.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2008). The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009a). The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009b). The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Schmelzeisen, L., Dima, C., and Staab, S. (2021). Wikidated 1.0: An evolving knowledge graph dataset of wikidata's revision history. In *Wikidata@ISWC*.

Schneider., P., Klettner., M., Jokinen., K., Simperl., E., and Matthes., F. (2024). Evaluating large language models in semantic parsing for conversational question answering over

knowledge graphs. In *Proceedings of the 16th International Conference on Agents and Artificial Intelligence - Volume 3: ICAART*, pages 807–814. INSTICC, SciTePress.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. (2017). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Shen, T., Geng, X., Qin, T., Guo, D., Tang, D., Duan, N., Long, G., and Jiang, D. (2019). Multi-task learning for conversational question answering over a large-scale knowledge base. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2442–2451, Hong Kong, China. Association for Computational Linguistics.

Shen, W., Wang, J., and Han, J. (2015). Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering*, 27(2):443–460.

Shrivastava, S. (2017). Bring rich knowledge of people, places, things and local businesses to your apps. *Bing Blogs*.

Singh, A. K. and Strouse, D. (2024). Tokenization counts: the impact of tokenization on arithmetic in frontier llms.

Singhal, A. (2012). Introducing the knowledge graph: things, not strings. *Google Blog*.

Srivastava, S., Azaria, A., and Mitchell, T. (2017). Parsing natural language conversations using contextual cues. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 4089–4095.

Suhr, A., Iyer, S., and Artzi, Y. (2018). Learning to map context-dependent sentences to executable formal queries. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2238–2249, New Orleans, Louisiana. Association for Computational Linguistics.

Sukhbaatar, S., szlam, a., Weston, J., and Fergus, R. (2015). End-to-end memory networks. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 28, pages 2440–2448. Curran Associates, Inc.

Sun, K., Qi, P., Zhang, Y., Liu, L., Wang, W., and Huang, Z. (2023). Tokenization consistency matters for generative models on extractive NLP tasks. In Bouamor, H., Pino, J., and Bali, K., editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 13300–13310, Singapore. Association for Computational Linguistics.

Sun, M. and Chai, J. Y. (2007a). Discourse processing for context question answering based on linguistic knowledge. *Knowledge-Based Systems*, 20(6):511–526.

Sun, M. and Chai, J. Y. (2007b). Discourse processing for context question answering based on linguistic knowledge. *Knowledge-Based Systems*, 20(6):511 – 526. Special Issue On Intelligent User Interfaces.

Talmor, A. and Berant, J. (2018). The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 641–651, New Orleans, Louisiana. Association for Computational Linguistics.

Taylor, W. L. (1953). "cloze procedure": A new tool for measuring readability. *Journalism Quarterly*, 30(4):415–433.

Team, G., Anil, R., Borgeaud, S., Wu, Y., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., et al. (2023). Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Team, G., Georgiev, P., Lei, V. I., Burnell, R., Bai, L., Gulati, A., Tanzer, G., Vincent, D., Pan, Z., Wang, S., Mariooryad, S., Ding, Y., Geng, X., Alcober, F., Frostig, R., Omernick, M., Walker, L., Paduraru, C., Sorokin, C., Tacchetti, A., Gaffney, C., Daruki, S., Sercinoglu, O., Gleicher, Z., Love, J., Voigtlaender, P., Jain, R., Surita, G., Mohamed, K., Blevins, R., Ahn, J., Zhu, T., Kawintiranon, K., Firat, O., Gu, Y., Zhang, Y., Rahtz, M., Faruqui, M., Clay, N., Gilmer, J., Co-Reyes, J., Penchev, I., Zhu, R., Morioka, N., Hui, K., Haridasan, K., Campos, V., Mahdieh, M., Guo, M., Hassan, S., Kilgour, K., Vezer, A., Cheng, H.-T., de Liedekerke, R., Goyal, S., Barham, P., Strouse, D., Noury, S., Adler, J., Sundararajan, M., Vikram, S., Lepikhin, D., Paganini, M., Garcia, X., Yang, F., Valter, D., Trebacz, M., Vodrahalli, K., Asawaroengchai, C., Ring, R., Kalb, N., Soares, L. B., Brahma, S., Steiner, D., Yu, T., Mentzer, F., He, A., Gonzalez, L., Xu, B., Kaufman, R. L., Shafey, L. E., Oh, J., Hennigan, T., van den

Driessche, G., Odoom, S., Lucic, M., Roelofs, B., Lall, S., Marathe, A., Chan, B., Ontanon, S., He, L., Teplyashin, D., Lai, J., Crone, P., Damoc, B., Ho, L., Riedel, S., Lenc, K., Yeh, C.-K., Chowdhery, A., Xu, Y., Kazemi, M., Amid, E., Petrushkina, A., Swersky, K., Khodaei, A., Chen, G., Larkin, C., Pinto, M., Yan, G., Badia, A. P., Patil, P., Hansen, S., Orr, D., Arnold, S. M. R., Grimstad, J., Dai, A., Douglas, S., Sinha, R., Yadav, V., Chen, X., Gribovskaya, E., Austin, J., Zhao, J., Patel, K., Komarek, P., Austin, S., Borgeaud, S., Friso, L., Goyal, A., Caine, B., Cao, K., Chung, D.-W., Lamm, M., Barth-Maron, G., Kagohara, T., Olszewska, K., Chen, M., Shivakumar, K., Agarwal, R., Godhia, H., Rajwar, R., Snaider, J., Dotiwalla, X., Liu, Y., Barua, A., Ungureanu, V., Zhang, Y., Batsaikhan, B.-O., Wirth, M., Qin, J., Danihelka, I., Doshi, T., Chadwick, M., Chen, J., Jain, S., Le, Q., Kar, A., Gurumurthy, M., Li, C., Sang, R., Liu, F., Lamprou, L., Munoz, R., Lintz, N., Mehta, H., Howard, H., Reynolds, M., Aroyo, L., Wang, Q., Blanco, L., Cassirer, A., Griffith, J., Das, D., Lee, S., Sygnowski, J., Fisher, Z., Besley, J., Powell, R., Ahmed, Z., Paulus, D., Reitter, D., Borsos, Z., Joshi, R., Pope, A., Hand, S., Selo, V., Jain, V., Sethi, N., Goel, M., Makino, T., May, R., Yang, Z., Schalkwyk, J., Butterfield, C., Hauth, A., Goldin, A., Hawkins, W., Senter, E., Brin, S., Woodman, O., Ritter, M., Noland, E., Giang, M., Bolina, V., Lee, L., Blyth, T., Mackinnon, I., Reid, M., Sarvana, O., Silver, D., Chen, A., Wang, L., Maggiore, L., Chang, O., Attaluri, N., Thornton, G., Chiu, C.-C., Bunyan, O., Levine, N., Chung, T., Eltyshev, E., Si, X., Lillicrap, T., Brady, D., Aggarwal, V., Wu, B., Xu, Y., McIlroy, R., Badola, K., Sandhu, P., Moreira, E., Stokowiec, W., Hemsley, R., Li, D., Tudor, A., Shyam, P., Rahimtoroghi, E., Haykal, S., Sprechmann, P., Zhou, X., Mincu, D., Li, Y., Addanki, R., Krishna, K., Wu, X., Frechette, A., Eyal, M., Dafoe, A., Lacey, D., Whang, J., Avrahami, T., Zhang, Y., Taropa, E., Lin, H., Toyama, D., Rutherford, E., Sano, M., Choe, H., Tomala, A., Safranek-Shrader, C., Kassner, N., Pajarskas, M., Harvey, M., Sechrist, S., Fortunato, M., Lyu, C., Elsayed, G., Kuang, C., Lottes, J., Chu, E., Jia, C., Chen, C.-W., Humphreys, P., Baumli, K., Tao, C., Samuel, R., dos Santos, C. N., Andreassen, A., Rakićević, N., Grewe, D., Kumar, A., Winkler, S., Caton, J., Brock, A., Dalmia, S., Sheahan, H., Barr, I., Miao, Y., Natsev, P., Devlin, J., Behbahani, F., Prost, F., Sun, Y., Myaskovsky, A., Pillai, T. S., Hurt, D., Lazaridou, A., Xiong, X., Zheng, C., Pardo, F., Li, X., Horgan, D., Stanton, J., Ambar, M., Xia, F., Lince, A., Wang, M., Mustafa, B., Webson, A., Lee, H., Anil, R., Wicke, M., Dozat, T., Sinha, A., Piqueras, E., Dabir, E., Upadhyay, S., Boral, A., Hendricks, L. A., Fry, C., Djolonga, J., Su, Y., Walker, J., Labanowski, J., Huang, R., Misra, V., Chen, J., Skerry-Ryan, R., Singh, A., Rijhwani, S., Yu, D., Castro-Ros, A., Changpinyo, B., Datta, R., Bagri, S., Hrafnkelsson,

A. M., Maggioni, M., Zheng, D., Sulsky, Y., Hou, S., Paine, T. L., Yang, A., Riesa, J., Rogozinska, D., Marcus, D., Badawy, D. E., Zhang, Q., Wang, L., Miller, H., Greer, J., Sjos, L. L., Nova, A., Zen, H., Chaabouni, R., Rosca, M., Jiang, J., Chen, C., Liu, R., Sainath, T., Krikun, M., Polozov, A., Lespiau, J.-B., Newlan, J., Cankara, Z., Kwak, S., Xu, Y., Chen, P., Coenen, A., Meyer, C., Tsihlas, K., Ma, A., Gottweis, J., Xing, J., Gu, C., Miao, J., Frank, C., Cankara, Z., Ganapathy, S., Dasgupta, I., Hughes-Fitt, S., Chen, H., Reid, D., Rong, K., Fan, H., van Amersfoort, J., Zhuang, V., Cohen, A., Gu, S. S., Mohananey, A., Ilic, A., Tobin, T., Wieting, J., Bortsova, A., Thacker, P., Wang, E., Caveness, E., Chiu, J., Sezener, E., Kaskasoli, A., Baker, S., Millican, K., Elhawaty, M., Aisopos, K., Lebsack, C., Byrd, N., Dai, H., Jia, W., Wiethoff, M., Davoodi, E., Weston, A., Yagati, L., Ahuja, A., Gao, I., Pundak, G., Zhang, S., Azzam, M., Sim, K. C., Caelles, S., Keeling, J., Sharma, A., Swing, A., Li, Y., Liu, C., Bostock, C. G., Bansal, Y., Nado, Z., Anand, A., Lipschultz, J., Karmarkar, A., Proleev, L., Ittycheriah, A., Yeganeh, S. H., Polovets, G., Faust, A., Sun, J., Rrustemi, A., Li, P., Shivanna, R., Liu, J., Welty, C., Lebron, F., Baddepudi, A., Krause, S., Parisotto, E., Soricut, R., Xu, Z., Bloxwich, D., Johnson, M., Neyshabur, B., Mao-Jones, J., Wang, R., Ramasesh, V., Abbas, Z., Guez, A., Segal, C., Nguyen, D. D., Svensson, J., Hou, L., York, S., Milan, K., Bridgers, S., Gworek, W., Tagliasacchi, M., Lee-Thorp, J., Chang, M., Guseynov, A., Hartman, A. J., Kwong, M., Zhao, R., Kashem, S., Cole, E., Miech, A., Tanburn, R., Phuong, M., Pavetic, F., Cevey, S., Comanescu, R., Ives, R., Yang, S., Du, C., Li, B., Zhang, Z., Iinuma, M., Hu, C. H., Roy, A., Bijwadia, S., Zhu, Z., Martins, D., Saputro, R., Gergely, A., Zheng, S., Jia, D., Antonoglou, I., Sadovsky, A., Gu, S., Bi, Y., Andreev, A., Samangooei, S., Khan, M., Kocisky, T., Filos, A., Kumar, C., Bishop, C., Yu, A., Hodkinson, S., Mittal, S., Shah, P., Moufarek, A., Cheng, Y., Bloniarz, A., Lee, J., Pejman, P., Michel, P., Spencer, S., Feinberg, V., Xiong, X., Savinov, N., Smith, C., Shakeri, S., Tran, D., Chesus, M., Bohnet, B., Tucker, G., von Glehn, T., Muir, C., Mao, Y., Kazawa, H., Slone, A., Soparkar, K., Shrivastava, D., Cobon-Kerr, J., Sharman, M., Pavagadhi, J., Araya, C., Misiunas, K., Ghelani, N., Laskin, M., Barker, D., Li, Q., Briukhov, A., Houlsby, N., Glaese, M., Lakshminarayanan, B., Schucher, N., Tang, Y., Collins, E., Lim, H., Feng, F., Recasens, A., Lai, G., Magni, A., Cao, N. D., Siddhant, A., Ashwood, Z., Orbay, J., Dehghani, M., Brennan, J., He, Y., Xu, K., Gao, Y., Saroufim, C., Molloy, J., Wu, X., Arnold, S., Chang, S., Schrittwieser, J., Buchatskaya, E., Radpour, S., Polacek, M., Giordano, S., Bapna, A., Tokumine, S., Hellendoorn, V., Sottiaux, T., Cogan, S., Severyn, A., Saleh, M., Thakoor, S., Shefey, L., Qiao, S., Gaba, M., yiin Chang, S., Swanson, C., Zhang, B., Lee, B., Rubenstein, P. K., Song,

G., Kwiatkowski, T., Koop, A., Kannan, A., Kao, D., Schuh, P., Stjerngren, A., Ghiasi, G., Gibson, G., Vilnis, L., Yuan, Y., Ferreira, F. T., Kamath, A., Klimenko, T., Franko, K., Xiao, K., Bhattacharya, I., Patel, M., Wang, R., Morris, A., Strudel, R., Sharma, V., Choy, P., Hashemi, S. H., Landon, J., Finkelstein, M., Jhakra, P., Frye, J., Barnes, M., Mauger, M., Daun, D., Baatarsukh, K., Tung, M., Farhan, W., Michalewski, H., Viola, F., de Chaumont Quitry, F., Lan, C. L., Hudson, T., Wang, Q., Fischer, F., Zheng, I., White, E., Dragan, A., baptiste Alayrac, J., Ni, E., Pritzel, A., Iwanicki, A., Isard, M., Bulanova, A., Zilka, L., Dyer, E., Sachan, D., Srinivasan, S., Muckenhirn, H., Cai, H., Mandhane, A., Tariq, M., Rae, J. W., Wang, G., Ayoub, K., FitzGerald, N., Zhao, Y., Han, W., Alberti, C., Garrette, D., Krishnakumar, K., Gimenez, M., Levskaya, A., Sohn, D., Matak, J., Iturrate, I., Chang, M. B., Xiang, J., Cao, Y., Ranka, N., Brown, G., Hutter, A., Mirrokni, V., Chen, N., Yao, K., Egyed, Z., Galilee, F., Liechty, T., Kallakuri, P., Palmer, E., Ghemawat, S., Liu, J., Tao, D., Thornton, C., Green, T., Jasarevic, M., Lin, S., Cotruta, V., Tan, Y.-X., Fiedel, N., Yu, H., Chi, E., Neitz, A., Heitkaemper, J., Sinha, A., Zhou, D., Sun, Y., Kaed, C., Hulse, B., Mishra, S., Georgaki, M., Kudugunta, S., Farabet, C., Shafran, I., Vlasic, D., Tsitsulin, A., Ananthanarayanan, R., Carin, A., Su, G., Sun, P., V, S., Carvajal, G., Broder, J., Comsa, I., Repina, A., Wong, W., Chen, W. W., Hawkins, P., Filonov, E., Loher, L., Hirnschall, C., Wang, W., Ye, J., Burns, A., Cate, H., Wright, D. G., Piccinini, F., Zhang, L., Lin, C.-C., Gog, I., Kulizhskaya, Y., Sreevatsa, A., Song, S., Cobo, L. C., Iyer, A., Tekur, C., Garrido, G., Xiao, Z., Kemp, R., Zheng, H. S., Li, H., Agarwal, A., Ngani, C., Goshvadi, K., Santamaria-Fernandez, R., Fica, W., Chen, X., Gorgolewski, C., Sun, S., Garg, R., Ye, X., Eslami, S. M. A., Hua, N., Simon, J., Joshi, P., Kim, Y., Tenney, I., Potluri, S., Thiet, L. N., Yuan, Q., Luisier, F., Chronopoulou, A., Scellato, S., Srinivasan, P., Chen, M., Koverkathu, V., Dalibard, V., Xu, Y., Saeta, B., Anderson, K., Sellam, T., Fernando, N., Huot, F., Jung, J., Varadarajan, M., Quinn, M., Raul, A., Le, M., Habalov, R., Clark, J., Jalan, K., Bullard, K., Singhal, A., Luong, T., Wang, B., Rajayogam, S., Eisenschlos, J., Jia, J., Finchelstein, D., Yakubovich, A., Balle, D., Fink, M., Agarwal, S., Li, J., Dvijotham, D., Pal, S., Kang, K., Konzelmann, J., Beattie, J., Dousse, O., Wu, D., Crocker, R., Elkind, C., Jonnalagadda, S. R., Lee, J., Holtmann-Rice, D., Kallarackal, K., Liu, R., Vnukov, D., Vats, N., Invernizzi, L., Jafari, M., Zhou, H., Taylor, L., Prendki, J., Wu, M., Eccles, T., Liu, T., Kopparapu, K., Beaufays, F., Angermueller, C., Marzoca, A., Sarcar, S., Dib, H., Stanway, J., Perbet, F., Trdin, N., Sterneck, R., Khorlin, A., Li, D., Wu, X., Goenka, S., Madras, D., Goldshtein, S., Gierke, W., Zhou, T., Liu, Y., Liang, Y., White, A., Li, Y., Singh, S., Bahargam, S., Epstein, M., Basu, S., Lao, L., Ozturel,

A., Crous, C., Zhai, A., Lu, H., Tung, Z., Gaur, N., Walton, A., Dixon, L., Zhang, M., Globerson, A., Uy, G., Bolt, A., Wiles, O., Nasr, M., Shumailov, I., Selvi, M., Piccinno, F., Aguilar, R., McCarthy, S., Khalman, M., Shukla, M., Galic, V., Carpenter, J., Villela, K., Zhang, H., Richardson, H., Martens, J., Bosnjak, M., Belle, S. R., Seibert, J., Alnahlawi, M., McWilliams, B., Singh, S., Louis, A., Ding, W., Popovici, D., Simicich, L., Knight, L., Mehta, P., Gupta, N., Shi, C., Fatehi, S., Mitrovic, J., Grills, A., Pagadora, J., Petrova, D., Eisenbud, D., Zhang, Z., Yates, D., Mittal, B., Tripuraneni, N., Assael, Y., Brovelli, T., Jain, P., Velimirovic, M., Akbulut, C., Mu, J., Macherey, W., Kumar, R., Xu, J., Qureshi, H., Comanici, G., Wiesner, J., Gong, Z., Ruddock, A., Bauer, M., Felt, N., GP, A., Arnab, A., Zelle, D., Rothfuss, J., Rosgen, B., Shenoy, A., Seybold, B., Li, X., Mudigonda, J., Erdogan, G., Xia, J., Simsa, J., Michi, A., Yao, Y., Yew, C., Kan, S., Caswell, I., Radebaugh, C., Elisseeff, A., Valenzuela, P., McKinney, K., Paterson, K., Cui, A., Latorre-Chimoto, E., Kim, S., Zeng, W., Durden, K., Ponnapalli, P., Sosea, T., Choquette-Choo, C. A., Manyika, J., Robenek, B., Vashisht, H., Pereira, S., Lam, H., Velic, M., Owusu-Afriyie, D., Lee, K., Bolukbasi, T., Parrish, A., Lu, S., Park, J., Venkatraman, B., Talbert, A., Rosique, L., Cheng, Y., Sozanschi, A., Paszke, A., Kumar, P., Austin, J., Li, L., Salama, K., Kim, W., Dukkipati, N., Baryshnikov, A., Kaplanis, C., Sheng, X., Chervonyi, Y., Unlu, C., de Las Casas, D., Askham, H., Tunyasuvunakool, K., Gimeno, F., Poder, S., Kwak, C., Miecnikowski, M., Mirrokni, V., Dimitriev, A., Parisi, A., Liu, D., Tsai, T., Shevlane, T., Kouridi, C., Garmon, D., Goedeckemeyer, A., Brown, A. R., Vijayakumar, A., Elqursh, A., Jazayeri, S., Huang, J., Carthy, S. M., Hoover, J., Kim, L., Kumar, S., Chen, W., Biles, C., Bingham, G., Rosen, E., Wang, L., Tan, Q., Engel, D., Pongetti, F., de Cesare, D., Hwang, D., Yu, L., Pullman, J., Narayanan, S., Levin, K., Gopal, S., Li, M., Aharoni, A., Trinh, T., Lo, J., Casagrande, N., Vij, R., Matthey, L., Ramadhana, B., Matthews, A., Carey, C., Johnson, M., Goranova, K., Shah, R., Ashraf, S., Dasgupta, K., Larsen, R., Wang, Y., Vuyyuru, M. R., Jiang, C., Ijazi, J., Osawa, K., Smith, C., Boppana, R. S., Bilal, T., Koizumi, Y., Xu, Y., Altun, Y., Shabat, N., Bariach, B., Korchemniy, A., Choo, K., Ronneberger, O., Iwuanyanwu, C., Zhao, S., Soergel, D., Hsieh, C.-J., Cai, I., Iqbal, S., Sundermeyer, M., Chen, Z., Bursztein, E., Malaviya, C., Biadsy, F., Shroff, P., Dhillon, I., Latkar, T., Dyer, C., Forbes, H., Nicosia, M., Nikolaev, V., Greene, S., Georgiev, M., Wang, P., Martin, N., Sedghi, H., Zhang, J., Banzal, P., Fritz, D., Rao, V., Wang, X., Zhang, J., Patraucean, V., Du, D., Mordatch, I., Jurin, I., Liu, L., Dubey, A., Mohan, A., Nowakowski, J., Ion, V.-D., Wei, N., Tojo, R., Raad, M. A., Hudson, D. A., Keshava, V., Agrawal, S., Ramirez, K., Wu, Z., Nguyen, H., Liu, J., Sewak, M., Petrini, B., Choi,

D., Philips, I., Wang, Z., Bica, I., Garg, A., Wilkiewicz, J., Agrawal, P., Li, X., Guo, D., Xue, E., Shaik, N., Leach, A., Khan, S. M., Wiesinger, J., Jerome, S., Chakladar, A., Wang, A. W., Ornduff, T., Abu, F., Ghaffarkhah, A., Wainwright, M., Cortes, M., Liu, F., Maynez, J., Terzis, A., Samangouei, P., Mansour, R., Kepa, T., Aubet, F.-X., Algymr, A., Banica, D., Weisz, A., Orban, A., Senges, A., Andrejczuk, E., Geller, M., Santo, N. D., Anklin, V., Merey, M. A., Baeuml, M., Strohman, T., Bai, J., Petrov, S., Wu, Y., Hassabis, D., Kavukcuoglu, K., Dean, J., and Vinyals, O. (2024). Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context.

Tjong Kim Sang, E. F. and Buchholz, S. (2000). Introduction to the CoNLL-2000 shared task chunking. In *Fourth Conference on Computational Natural Language Learning and the Second Learning Language in Logic Workshop*.

Tobin, F. (2017). Thomson reuters launches first of its kind knowledge graph feed allowing financial services customers to accelerate their ai and digital strategies. *Thomson Reuters Press Release*.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. (2023a). Llama: Open and efficient foundation language models.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. (2023b). Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017a). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017b). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph attention networks. In *6th International Conference on Learning Representa-*

tions, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph attention networks. In *International Conference on Learning Representations*.

von Werra, L., Belkada, Y., Tunstall, L., Beeching, E., Thrush, T., Lambert, N., and Huang, S. (2020). Trl: Transformer reinforcement learning. `https://github.com/huggingface/trl`.

Voorhees, E. M. (2004). Overview of TREC 2004. In Voorhees, E. M. and Buckland, L. P., editors, *Proceedings of the Thirteenth Text REtrieval Conference, TREC 2004, Gaithersburg, Maryland, USA, November 16-19, 2004*, volume 500-261 of *NIST Special Publication*. National Institute of Standards and Technology (NIST).

Voskarides, N., Li, D., Ren, P., Kanoulas, E., and de Rijke, M. (2020). Query resolution for conversational search with limited supervision. In *SIGIR*.

Vrandečić, D. and Krötzsch, M. (2014). Wikidata: A free collaborative knowledge base. *CACM*.

Vrandečić, D. (2012). Wikidata: A new platform for collaborative data collection. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12 Companion, page 1063–1064, New York, NY, USA. Association for Computing Machinery.

Vrandečić, D. and Krötzsch, M. (2014). Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Wang, B., Shin, R., Liu, X., Polozov, O., and Richardson, M. (2020). RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578, Online. Association for Computational Linguistics.

Wang, H., Feng, S., He, T., Tan, Z., Han, X., and Tsvetkov, Y. (2024). Can language models solve graph problems in natural language? *Advances in Neural Information Processing Systems*, 36.

Wang, T., Roberts, A., Hesslow, D., Scao, T. L., Chung, H. W., Beltagy, I., Launay, J., and Raffel, C. (2022). What language model architecture and pretraining objective works

best for zero-shot generalization? In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S., editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 22964–22984. PMLR.

Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., and Hajishirzi, H. (2023). Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., and Fedus, W. (2022a). Emergent abilities of large language models. *Transactions on Machine Learning Research*. Survey Certification.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., ichter, b., Xia, F., Chi, E., Le, Q. V., and Zhou, D. (2022b). Chain-of-thought prompting elicits reasoning in large language models. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Weston, J., Chopra, S., and Bordes, A. (2015). Memory networks. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. (2020). Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Wu, Q., Bansal, G., Zhang, J., Wu, Y., Zhang, S., Zhu, E., Li, B., Jiang, L., Zhang, X., and Wang, C. (2023). Autogen: Enabling next-gen llm applications via multi-agent conversation framework. *arXiv preprint arXiv:2308.08155*.

Xie, J., Zhang, K., Chen, J., Zhu, T., Lou, R., Tian, Y., Xiao, Y., and Su, Y. (2024a). TravelPlanner: A benchmark for real-world planning with language agents. In Salakhut-

dinov, R., Kolter, Z., Heller, K., Weller, A., Oliver, N., Scarlett, J., and Berkenkamp, F., editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 54590–54613. PMLR.

Xie, T., Zhou, F., Cheng, Z., Shi, P., Weng, L., Liu, Y., Hua, T. J., Zhao, J., Liu, Q., Liu, C., Liu, Z., Xu, Y., SU, H., Shin, D., Xiong, C., and Yu, T. (2024b). Openagents: An open platform for language agents in the wild. In *First Conference on Language Modeling*.

Xu, S., Liu, S., Culhane, T., Pertseva, E., Wu, M.-H., Semnani, S., and Lam, M. (2023). Fine-tuned LLMs know more, hallucinate less with few-shot sequence-to-sequence semantic parsing over Wikidata. In Bouamor, H., Pino, J., and Bali, K., editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5778–5791, Singapore. Association for Computational Linguistics.

Ye, R., Zhang, C., Wang, R., Xu, S., and Zhang, Y. (2023). Natural language is all a graph needs. *arXiv preprint arXiv:2308.07134*.

Yin, P., Li, W.-D., Xiao, K., Rao, A., Wen, Y., Shi, K., Howland, J., Bailey, P., Catasta, M., Michalewski, H., Polozov, O., and Sutton, C. (2023). Natural language to code generation in interactive data science notebooks. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 126–173, Toronto, Canada. Association for Computational Linguistics.

Yin, X., Gromann, D., and Rudolph, S. (2021). Neural machine translating from natural language to SPARQL. *Future Generation Computer Systems*, 117:510–519.

You, Z., Zhang, Y., Xu, D., Lou, Y., Yan, Y., Wang, W., Zhang, H., and Huang, Y. (2025). Datawiseagent: A notebook-centric llm agent framework for automated data science. *arXiv preprint arXiv:2503.07044*.

Yu, T., Zhang, R., Er, H., Li, S., Xue, E., Pang, B., Lin, X. V., Tan, Y. C., Shi, T., Li, Z., Jiang, Y., Yasunaga, M., Shim, S., Chen, T., Fabbri, A., Li, Z., Chen, L., Zhang, Y., Dixit, S., Zhang, V., Xiong, C., Socher, R., Lasecki, W., and Radev, D. (2019a). CoSQL: A conversational text-to-SQL challenge towards cross-domain natural language interfaces to databases. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1962–1979, Hong Kong, China. Association for Computational Linguistics.

Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., Ma, J., Li, I., Yao, Q., Roman, S., Zhang, Z., and Radev, D. (2018). Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

Yu, T., Zhang, R., Yasunaga, M., Tan, Y. C., Lin, X. V., Li, S., Er, H., Li, I., Pang, B., Chen, T., Ji, E., Dixit, S., Proctor, D., Shim, S., Kraft, J., Zhang, V., Xiong, C., Socher, R., and Radev, D. (2019b). SParC: Cross-domain semantic parsing in context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4511–4523, Florence, Italy. Association for Computational Linguistics.

Zamani, H., Trippas, J. R., Dalton, J., and Radlinski, F. (2022). Conversational information seeking. https://arxiv.org/abs/2201.08808.

Zelle, J. M. and Mooney, R. J. (1996). Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, AAAI'96, page 1050–1055. AAAI Press.

Zettlemoyer, L. and Collins, M. (2009). Learning context-dependent mappings from sentences to logical form. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 976–984, Suntec, Singapore. Association for Computational Linguistics.

Zhang, H., Cao, R., Chen, L., Xu, H., and Yu, K. (2023). ACT-SQL: In-context learning for text-to-SQL with automatically-generated chain-of-thought. In Bouamor, H., Pino, J., and Bali, K., editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3501–3532, Singapore. Association for Computational Linguistics.

Zhang, H., Cao, R., Xu, H., Chen, L., and Yu, K. (2024a). CoE-SQL: In-context learning for multi-turn text-to-SQL with chain-of-editions. In Duh, K., Gomez, H., and Bethard, S., editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6487–6508, Mexico City, Mexico. Association for Computational Linguistics.

Zhang, R., Yu, T., Er, H., Shim, S., Xue, E., Lin, X. V., Shi, T., Xiong, C., Socher, R., and Radev, D. (2019). Editing-based SQL query generation for cross-domain

context-dependent questions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5338–5349, Hong Kong, China. Association for Computational Linguistics.

Zhang, W., Shen, Y., Lu, W., and Zhuang, Y. (2024b). Data-copilot: Bridging billions of data and humans with autonomous workflow. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*.

Zhong, V., Xiong, C., and Socher, R. (2018). Seq2SQL: Generating structured queries from natural language using reinforcement learning.

Zhou, S., Xu, F. F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., Cheng, X., Ou, T., Bisk, Y., Fried, D., et al. (2023). Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*.