QUESTION ANSWERING AND GENERATION FROM STRUCTURED AND UNSTRUCTURED DATA

Yu Chen

Submitted in Partial Fullfillment of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

Approved by:
Dr. Mohammed J. Zaki, Chair
Dr. Bulent Yener
Dr. Alex Gittens
Dr. Qiang Ji
Dr. Lingfei Wu

Department of Computer Science
Rensselaer Polytechnic Institute
Troy, New York

[August 2020]
Submitted June 2020

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENT

First of all, I would like to express my sincere gratitude and appreciation to my adviser Prof. Mohammed J. Zaki for always being so inspirational and supportive during my doctoral studies. He is not only an excellent mentor with stringent academic attitude, rich source of expertise and broad research vision, but also a very nice person who is always ready to listen to my opinions and will respect them. I have often been impressed and encouraged by his great passion for research as well as his creative ideas and remarkable insights. The body of work presented in this dissertation would never have been possible without his continuous guidance and support.

I would also like to sincerely thank the rest of my committee members Prof. Bulent Yener, Prof. Alex Gittens, Prof. Qiang Ji and Dr. Lingfei Wu for their valuable time, effort and insightful comments. Special thanks to Dr. Lingfei Wu for being such a great collaborator and mentor. I have always enjoyed many pleasant and fruitful discussions with him from which I have learned a lot. Prof. Qiang Ji taught a great deep learning course which started my journey into Deep Learning. I learned a lot from the Computer Science courses in our department. In this regard, I would like to thank Prof. Bulent Yener and Prof. Alex Gittens.

Also I want to thank my current and former colleagues at RPI: Dr. Oshani W. Seneviratne, Dr. Nidhi Rastogi, Yuchen Liang, Diya Li, Ananya Subburathinam, Vipula Rawte, Jon Harris, Steven Haussmann, and Dylan Elliott. We have had many nice discussions and I am glad to have had the opportunity to work with most of them on the same project. In this regard, I would like to thank my current and former IBM colleagues in the HEALS project: Dr. Ching-Hua Chen, Dr. James Codella, Chandramouli Maduri, Dr. Sun Si and Dr. Kim Walter. In addition, I have to thank Dr. Lifu Huang from UIUC and Dr. Liu Yang from UMass (now at Google) who have provided valuable suggestions at important moments during my doctoral studies.

I am indebted to Dr. Lazaros Polymenakos who was my manager during my internship. I spent a wonderful summer in Yorktown Heights working at IBM Research. Thanks are also due to my other IBM colleagues during the internship.

I am thankful to Tracy Hoffman, Shannon Carrothers, Chris Coonrad and Terry Hayden, who have been or had been providing administrative support and organizing depart-

ment activities constantly. Also thanks to David Blevins, Steven Lindsey and Peter Bailie for maintaining the server clusters and kindly helping me take advantage of the available computation power.

I would also like to take this opportunity to express my sincere appreciation to my dear girlfriend, Yuwei Guo, for her continuous love, support, and affection.

Last but not the least, my genuine gratefulness must go to my parents who raised me up and always support me with all their heart. I appreciate my parents for all their sacrifice and support on my education and pursuit of my dream. They are the most important people in my world and I dedicate this thesis to them.

ABSTRACT

This dissertation focuses on two different but related natural language processing problems, namely, question answering and question generation. Automated question answering (QA) is the process of finding answers to natural language questions. This problem has been studied since the early days of artificial intelligence and recently has drawn increasing attention. Automatic question answering usually relies on some underlying knowledge sources, e.g., a knowledge base (KB), a database or just free-form text from which answers can be gleaned. As such question answering has numerous applications in areas such as natural language interfaces to databases and spoken dialog systems.

We identify in particular three main challenges of question answering. The first challenge is the lexical gap between the questions and the underlying knowledge source. Human language is very flexible. The same question can be expressed in various ways while the knowledge source may use a canonical lexicon. It is therefore nontrivial to map a natural language question to the knowledge source. The second challenge is the problem of complex reasoning in question answering. Many realistic questions are complex since they require multi-hop reasoning. In order to answer those complex questions, an agent typically needs to perform a series of discrete symbolic operations such as arithmetic operations, logical operations, quantitative operations and comparative operations. Making machines perform complex reasoning automatically is fundamentally challenging. Even though one can predefine a set of discrete operations an agent can take, the program search space is still very large because of combinatorial explosion. The third challenge is conversational question answering. In real world scenarios, most questions are asked in a conversational manner. It is quite often that a question is asked within a certain conversational context. In other words, in a conversation, sequential questions are asked and a question being asked at a given turn might refer back to some previous questions or answers. Conversational question answering is significantly more challenging than single-turn question answering since the conversation history must be taken into account effectively in order to understand the question correctly.

Natural question generation (QG) is the task of generating natural language questions from a given form of data such as text, images, tables and knowledge graphs (KGs). As a dual task of question answering, question generation has many useful applications such as improving the question answering task by providing more training data, generating practice

exercises and assessments for educational purposes, and helping dialog systems to kick-start and continue a conversation with human users.

We identify in particular three main challenges of question generation. The first challenge is how to effectively model the context information (e.g., text and KGs). It is extremely important for a QG system to well understand the semantic meanings of the context so as to generate high-quality questions. Particularly, it becomes challenging to model long/large context with rich structure information. The second challenge is how to effectively leverage the answer information for question generation. Answer information is crucial for generating relevant and high quality questions because it can serve as a guidance on "what to ask" from the given context. However, most existing methods do not fully utilize the answer information when generating questions. The third challenge is how to effectively optimize a sequence learning model. Cross-entropy loss is widely used for training sequence learning neural networks. However, it has been observed that optimizing cross-entropy based training objectives for sequence learning does not always produce the best results on discrete evaluation metrics. Major limitations of this strategy include exposure bias and evaluation discrepancy between training and testing.

In this dissertation, we propose novel and effective approaches to address the above challenges of QA and QG tasks. On the QA side, we first propose a modular deep learning approach to automatically answer natural language questions over a large-scale knowledge base. Specifically, we propose to directly model the two-way flow of interactions between the questions and the underlying KB. The proposed model is able to perform multi-hop reasoning in a KB and requires no external resources and very few hand-crafted features. We show that on a popular WebQuestions KBQA benchmark, our model significantly outperforms previous information retrieval based methods while remaining competitive with handcrafted semantic parsing based methods. Then, we present a novel graph neural network (GNN) based model which is able to capture conversational flow in a dialog when executing the task of conversational machine reading comprehension where the knowledge source is free-form text. Based on the proposed Recurrent Graph Neural Network, we introduce a flow mechanism to model the temporal dependencies in a sequence of context graphs which represent the conversational history. On three public benchmarks, the proposed model shows superior performance compared to existing state-of-the-art methods. In addition, visualization experiments show that our model can offer good interpretability for the reasoning process.

On the QG side, we first present a novel bidirectional graph-to-sequence model for the task of QG from KGs. Specifically, we propose to apply a bidirectional graph-to-sequence model to encode the KG subgraph. Furthermore, we enhance our Recurrent Neural Network (RNN) based decoder with the novel node-level copying mechanism to allow directly copying node attributes from the KG subgraph to the output question. Both automatic and human evaluation results demonstrate that our model achieves new state-of-the-art scores, outperforming existing methods by a significant margin on two QG benchmarks. Experiments also show that our QG model can consistently benefit the QA task as a mean of data augmentation. Then, we propose a reinforcement learning (RL) based graph-to-sequence model for the task of QG from text. Our model consists of a graph-to-sequence generator with a novel Bidirectional Gated Graph Neural Network based encoder to embed the passage, and a hybrid evaluator with a mixed objective combining both the cross-entropy loss and the RL loss to ensure the generation of syntactically and semantically valid text. We also introduce an effective Deep Alignment Network for incorporating the answer information into the passage at both the word and contextual levels. Our model is end-to-end trainable and achieves new state-of-the-art scores, outperforming existing methods by a significant margin on the standard SQuAD benchmark.

CHAPTER 1
INTRODUCTION

1.1 Question Answering

1.1.1 Background

1.1.1.1 *A Brief History of Question Answering*

Automated question answering (QA) is the task of answering natural language questions using certain knowledge sources. It has been explored since the early days of artificial intelligence in many areas of Natural Language Processing (NLP) research. Natural language interfaces to databases (NLIDBs) allow users to access information stored in a database via natural language queries, instead of restricted SQL queries, which can make the database system more user-friendly. First attempts for NLIDBs appeared as early as the sixties. LUNAR [1] was designed to answer natural language questions about the geological analysis of lunar rocks returned by the Apollo missions. This research area became very popular in the mid-eighties. Despite the interest and numerous attempts, NLIDBs did not gain the expected rapid and wide commercial acceptance due to their poor performance and the difficulty for porting and configuring them. Question answering has also been an important topic in spoken dialog systems, which aim to enable computer systems to converse with a human with voice.

Since the nineties, researchers have showed great interest in open domain question answering. [2] designed a system called MURAX to answer general-knowledge questions using an on-line encyclopedia. Since 1999, the Text Retrieval Conference (TREC) has hosted QA competition tracks every year to advance the research on open-domain question answering.

Portions of this chapter previously appeared as: Y. Chen, L. Wu, and M. J. Zaki, "Bidirectional attentive memory networks for question answering over knowledge bases," in *Proc. 2019 Conf. N. Amer. Chap. Assoc. Comput. Ling.: Human Lang. Technol.*, vol. 1, Jun. 2-7, 2019, pp. 2913–2923.

Portions of this chapter previously appeared as: Y. Chen, L. Wu, and M. J. Zaki, "Graphflow: Exploiting conversation flow with graph neural networks for conversational machine comprehension," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Jul. 2020, pp. 1230–1236. Copyright © 2020, IJCAI (`https://www.ijcai.org`).

Portions of this chapter have been submitted to: Y. Chen, L. Wu, and M. J. Zaki, "Toward subgraph guided knowledge graph question generation with graph neural networks," in *Proc. 2020 Conf. Empirical Meth. Natural Lang. Process.*

Portions of this chapter previously appeared as: Y. Chen, L. Wu, and M. J. Zaki, "Reinforcement learning based graph-to-sequence model for natural question generation," in *Proc. 8th Int. Conf. Learn. Representations*, Apr. 26-30, 2020. [Online]. Available: https://openreview.net/forum?id=HygnDhEtvr

With the rapid evolution of deep learning, recent attempts [3]–[10] focus on applying deep learning approaches to this old yet very challenging research area.

1.1.1.2 *Open-domain vs. Closed-domain Question Answering*

According to whether the types of questions accepted are limited or not, we can generally categorize QA systems into two camps: open-domain question answering and closed-domain question answering.

Open-domain question answering deals with all kinds of questions about any topic. In other words, we do not restrict the types of questions which can be asked and hence the knowledge needed to answer those questions. Building such a QA system can be extremely hard since the system is supposed to be generic enough to be able to answer all kinds of questions and also scalable enough to be able to process potentially the whole world knowledge.

Unlike open-domain question answering that is ambitious, closed-domain question answering deals with questions under a specific domain, e.g., weather, healthcare or travel. Due to this restricted setting, a QA system can expect that the types of questions that will be asked as well as the knowledge needed to answer those questions are limited and known beforehand. This makes the problem more tractable compared to open-domain question answering, as one can exploit domain-specific prior knowledge when designing the system so as to make it work reasonably well for a specific domain. Nowadays, almost all successful commercial QA products such as Amazon Alexa, Google Home and Apple HomePod are essentially closed-domain QA systems, though the domains are continually expanding.

1.1.1.3 *Structured vs. Unstructured Knowledge Sources*

The world knowledge can be stored in very different forms. We can generally classify knowledge sources into two categories: structured knowledge sources and unstructured knowledge sources. Here we discuss in particular the following three types of knowledge sources which we will cover in this dissertation.

Free-form text is the most common and prevalent form of knowledge source. However, it is unstructured and hence not machine-readable.

Databases are also widely used to store knowledge and they are structured. A database is an organized collection of data, generally stored and accessed electronically from a computer

system. Relational database systems model data as rows and columns in a series of tables.

Knowledge bases (KBs) are a special kind of database for knowledge management. They provide the means for the computerized collection, organization, and retrieval of knowledge. A KB basically contains a large number of triples, where each triple consists of a subject, a relation and an object. A KB can store a lot of entities and the relations among those entities. Over the past decade, many large-scale open-domain knowledge bases (KBs) such as DBPedia [11], FreeBase [12], Yago3 [13] and WikiData [14] have been created and some of them are still growing.

Accordingly, we can categorize QA systems into different camps based on their knowledge sources, such as Knowledge Base Question Answering (KBQA) which uses a KB as knowledge source, Table-based Question Answering (TBQA) which uses a database as knowledge source and Machine Reading Comprehension (MRC) which uses free-form text as knowledge source.

Notably, one can design QA systems using hybrid knowledge sources. As such an open-domain QA system should take advantage of as many knowledge sources as possible so as to have a great coverage of world knowledge.

1.1.2 Motivation

Question answering finds several compelling applications, such as those briefly described next.

Natural language interfaces to databases. Natural language interfaces to databases (NLIDBs) aim to directly query a database system via natural language instead of some structured logic forms such as SQL. Compared to SQL, natural language queries are more flexible and user-friendly, because users do not need to have any knowledge about database systems or the schemas of the databases they are accessing. The underlying QA system is able to handle everything from natural language understanding to query execution.

Spoken dialog systems. A spoken dialog system is another application area of automated question answering. One of the most crucial parts in human-machine conversation is actually question answering. Virtual assistants such as Amazon Alexa provide many services to humans by answering questions.

Beyond search engines. While modern search engines such as Google can provide users a bunch of relevant documents which might contain the answers to their questions, users still

have to dig into a few documents at the top of the list in order to get the exact answers. Automated question answering goes one step further by providing users the exact answers they are looking for using various kinds of resources.

Adaptive education. Automated question answering can also be applied for adaptive education. Learners can learn knowledge and skills by asking an agent questions and getting answers from the agent.

1.1.3 Challenges

1.1.3.1 *Lexical Gap*

Human language is very flexible. One big challenge for questing answering is the lexical gap between the questions and the underlying knowledge source. The same question can be expressed in various ways in NL while the knowledge source may use another lexicon. It is therefore nontrivial to map an NL question to the knowledge source. For instance, it can be difficult for machines to map a question *Who is the wife of President Obama?* to a triple [*Barack Obama, married to, Michelle Obama*]. In order to do this, the system must understand "wife" is semantically related to "marry" and "President Obama" is an alias of "Barack Obama". However, most existing embedding-based methods [5], [9], [15]–[18] for question answering ignore the subtle inter-relationships between the question and the knowledge source.

1.1.3.2 *Complex Reasoning*

Many realistic questions are complex since they require multi-hop reasoning. In order to answer those complex questions, an agent typically needs to perform a series of discrete symbolic operations such as arithmetic operations, logical operations, quantitative operations and comparative operations. Recent years have observed a surge of interest in complex question answering. Many benchmarks have been released over the past few years for complex KBQA [19]–[22], MRC [23] and TBQA [24], [25]. Table 1.1 shows the various types of complex KBQA questions that the CSQA dataset [19] explored.

Making machines perform complex reasoning automatically is fundamentally challenging. Even though one can predefine a set of discrete operations an agent can take, the program search space is still quite large because of combinatorial explosion. Here, we briefly cover two main research streams for tackling this problem: symbolic AI and connectionist

Table 1.1: Types of complex questions.

Reasoning	Type	Example
Logical	Union	Which rivers flow through India or China?
	Intersection	Which rivers flow through India and China?
	Difference	Which rivers flow through India but not China?
Verification	Boolean	Does the Hudson River flow through New York City?
Quantitative	Count	How many revers flow through the USA?
	Min/Max	Which city in China has maximum number of rivers?
	Atleast/Atmost	Which country has at least N cities?
Comparative	More/Less	Which countries have more people than Japan?
	Count over More/Less	How many countries have more people than Japan?

AI, which can help us better understand why this is a challenging problem.

Symbolic AI. Symbolic approaches are based on high-level symbolic representations of problems, logic and search, and thus are efficient in execution of discrete operations. However, they either heavily rely on predefined rules or are very difficult to train especially at initial stages.

Connectionist AI. Deep neural networks-based approaches can be trained in an end-to-end fashion and have shown great promise in many question answering tasks [6], [8]. However, they are notorious for their incapability of executing symbolic operations and lack of explicit interpretability.

In Section 2.1.3, we will discuss recent attempts [6], [7], [25]–[32] on solving the complex reasoning problem in question answering and we will see that many of them are trying to combine the benefits of the above two camps.

1.1.3.3 *Conversational Question Answering*

In real world scenarios, most questions are asked in a conversational manner. It is quite often that a question is asked within a certain conversational context. In other words, in a conversation, sequential questions are asked and a question being asked at a certain turn might refer back to some previous questions or answers, typically by coreference or ellipsis. Recently, many conversational question answering benchmarks have been proposed for KBQA [19], [33], TBQA [34]–[36] and MRC [37]–[40].

Conversational question answering is significantly more challenging than single-turn question answering since conversation history must be taken into account effectively in order to understand the question correctly. In those human-to-human conversations [19], [33], [37]–

[39], the focus often shifts as the conversation progresses and linguistic phenomena such as coreference and ellipsis happen a lot. In order to understand the question being asked, a QA system must effectively utilize conversation history. However, most existing approaches [41]–[43] do not effectively capture conversation history and thus have trouble handling questions involving coreference or ellipsis.

1.2 Question Generation

1.2.1 Background

1.2.1.1 *A Brief History of Question Generation*

Natural question generation (QG) is the task of generating natural language questions from a given form of data. Traditional QG mainly focused on generating factoid questions from a single sentence or a paragraph [44]–[52]. Recent works started to explore other forms of knowledge sources such as images [53]–[58], tables [59]–[62], and knowledge graphs (KGs) [63]–[69]. However, early works [63]–[65], [70]–[72] on QG relied on template-based approaches that require significant amount of human effort, and thus have low generalizability and scalability. Encouraged by the huge success of Neural Networks (NNs), recent attempts [66]–[68], [73]–[76] have been focused on exploiting Neural Network (NN) based approaches that do not require manually-designed rules and are end-to-end trainable. Specifically, most of them formulate the QG task as a sequence-to-sequence (Seq2Seq) learning problem [77], [78]. This Seq2Seq paradigm has proven useful in various NLP tasks such as machine translation [77]–[79], text summarization [41], [80], [81] and dialog systems [82], [83].

1.2.1.2 *Answer-agnostic vs. Answer-aware Question Generation*

Based on whether the target answer information is utilized by a QG system or not, research on question generation can be broadly categorized into two classes: answer-agnostic QG and answer-aware QG. Early works [64], [66], [73], [84] on QG mainly focused on the answer-agnostic setting where an input context is given, and the goal is to generate a natural language question that is meaningful and relevant to the input context. The limitation of the answer-agnostic setting is that there is no control about which part of the context the generated question is asking about. Recent works explored the answer-aware setting where the target answer is either provided as input [69], [85]–[91] or extracted from the input

context by the system [67], [74], [76], [92]–[96].

1.2.1.3 *Structured vs. Unstructured Knowledge Sources*

The knowledge sources (i.e., context) of a QG system can either be structured or unstructured. While it seems straightforward to apply Recurrent Neural Networks (RNNs) or Convolutional Neural Networks (CNNs) to unstructured data such as text or images [44]–[58], it needs more thoughtful consideration on how to model structured data such as tables and KGs [59]–[69]. It is important to capture the structure information (e.g., relationships among objects in the strcutured data) in order to well understand the semantic meanings of the structured data.

1.2.2 Motivation

Natural question generation has many useful applications such as improving the question answering task [42], [97]–[100] by providing more training data [86], [92], [94], [101]–[103], generating practice exercises and assessments for educational purposes [71], [104], [105], and helping dialog systems to kick-start and continue a conversation with human users [54].

1.2.3 Challenges

1.2.3.1 *Context Modeling*

As mentioned earlier, QG aims to generate natural language questions from certain form of context (e.g., text, images, tables and KGs). In order to generate meaningful and relevant questions, it is essential to well understand the context information which can be represented in various forms. Here we would like to highlight two of the challenges regarding context modeling.

Modeling long/large context. Most existing works on QG mainly focused on using short/small context when generating questions. For example, existing models on QG from text mainly focused on using short passages (i.e., one or two sentences) as the input, which might limit the model capacity of generating complex questions requiring multi-hop reasoning. However, long text has posed challenges for existing QG models because i) it is usually more challenging to model long text for RNN-based neural models, and ii) selecting useful cues and relevant content from long text for QG is more difficult compared to short text. Recent works [87], [95], [106], [107] started to explore QG from long text via various hierar-

chical and attention models. This observation also holds true for the task of QG from KGs. Existing works on QG from KGs [66]–[68] only focused on generating simple questions from a single triple, which is not realistic in practice. Recently, [69] proposed a Transformer [108] based encoder-decoder model for QG from a KG subgraph.

Modeling structure information in context. Modeling structure information contained in context is also challenging and less studied in the field of QG. For example, existing works on QG from text [73]–[76] typically ignore the hidden structural information associated with a word sequence such as the syntactic parsing tree. Failing to utilize the rich text structure information beyond the simple word sequence may limit the effectiveness of these models for QG. Similarly, existing works [66]–[68] on QG from KGs typically employ an RNN-based encoder which cannot handle graph-structured input data.

1.2.3.2 *Utilizing Answer Information*

Utilizing answer information is crucial for generating high-quality questions, especially when the input context is long or large. When question generation is guided by the semantics of an answer, the resulting questions become more relevant and readable. Early works [73], [84] did not take into account the answer information when generating a question. Recent works [74], [87]–[91] have started to explore various means of utilizing the answer information such as by simply marking the answer location in the context [74], [87], [88], or using complex context-answer matching strategies [89], or separating answers from context when applying an encoder-decoder model [90], [91]. However, they neglect potential semantic relations between context words and answer words, and thus fail to explicitly model the global interactions among them in the embedding space.

1.2.3.3 *Training Sequence Learning Models*

It has been observed that in general, cross-entropy based sequence training has several limitations like exposure bias and inconsistency between train/test measurement [109], [110]. That is to say, in the training phase, a model has access to the ground-truth previous token when decoding and is optimized toward cross-entropy loss, while in the testing phase, no ground-truth previous token is provided and cross-entropy loss is not used for evaluation. As a result, they do not always produce the best results in terms of discrete evaluation metrics on sequence generation tasks such as text summarization [81] or question generation [89].

To cope with these issues, some recent QG approaches [84], [89] directly optimize evaluation metrics using Reinforcement Learning (RL) [111]. However, existing approaches usually only employ evaluation metrics like BLEU and ROUGE-L as rewards for RL training. More importantly, they fail to exploit other important metrics such as syntactic and semantic constraints for guiding high-quality text generation.

1.2.3.4 *Evaluation*

It is challenging to evaluate a question generation system, which is also true for any natural language generation (NLG) system. Generally speaking, syntactically correct, semantically sound, meaningful and natural questions are all useful evaluation criteria, yet they are hard to quantify [112]. In practice, human evaluation is usually conducted by randomly sampling a few hundred generated questions, and asking human evaluators to rate them based on some evaluation criteria. Because human evaluation is time-consuming and requires a lot of human effort, common automatic evaluation metrics for NLG, such as BLEU-4 [113], METEOR [114] and ROUGE-L [115], are also widely used.

However, some studies [116], [117] have shown that these automatic metrics do not correlate well with adequacy, fluency and coherence, as they essentially compute the n-gram similarity between the reference sentence and the generated sentence. To overcome this in the literature of QG evaluation, [118] proposed a new metric to evaluate the "answerability" of a question by calculating the scores for several question-specific factors, including question type, content words, function words, and named entities. The introduced Q-BLEU1 score was shown to correlate significantly better with human judgment compared to existing automatic metrics. Recent works [119]–[121] proposed similarity-based evaluation metrics by leveraging contextualized embeddings learned by large-scale pretrained language models such as ELMO [122] and BERT [123]. These automatic metrics reportedly show a high correlation with human judgment of text quality.

1.3 Contributions

This dissertation focuses on two different but related natural language processing problems, namely, question answering and question generation. We explore the multiple dimensions (e.g., knowledge sources, reasoning complexity and conversational property) of the question answering and generation tasks, and design novel and effective deep learning ap-

proaches for both tasks.

Knowledge base question answering. In Chapter 3, we describe a modular deep learning approach to automatically answer natural language questions over a large-scale knowledge base. Specifically, we propose to directly model the two-way flow of interactions between the questions and the underlying KB. The proposed model is able to perform multi-hop reasoning in a knowledge base and requires no external resources and very few hand-crafted features. We show that our model significantly outperforms previous information retrieval based methods while remaining competitive with handcrafted semantic parsing based methods on a popular KBQA benchmark.

Conversational machine reading comprehension. In Chapter 4, we propose a novel graph neural network based model which is able to capture conversational flow in a dialog when executing the task of conversational machine reading comprehension. Based on the proposed Recurrent Graph Neural Network, we introduce a flow mechanism to model the temporal dependencies in a sequence of context graphs which represent the conversational history. On three public benchmarks, the proposed model shows superior performance compared to existing state-of-the-art methods. In addition, visualization experiments show that our model can offer good interpretability for the reasoning process.

Natural question generation from KGs. In Chapter 5, we apply a novel bidirectional Graph2Seq model to encode the KG subgraph. Furthermore, we enhance our RNN decoder with the novel node-level copying mechanism to allow directly copying node attributes from the KG subgraph to the output question. Both automatic and human evaluation results demonstrate that our model achieves new state-of-the-art scores, outperforming existing methods by a significant margin on two QG benchmarks. Experiments also show that our QG model can consistently benefit the QA task as a mean of data augmentation.

Natural question generation from text. In Chapter 6, we propose a reinforcement learning (RL) based Graph2Seq model for QG from text. Our model consists of a Graph2Seq generator with a novel Bidirectional Gated Graph Neural Network based encoder to embed the passage, and a hybrid evaluator with a mixed objective combining both cross-entropy and RL losses to ensure the generation of syntactically and semantically valid text. We also introduce an effective Deep Alignment Network for incorporating the answer information into the passage at both the word and contextual levels. Our model is end-to-end trainable and achieves new state-of-the-art scores, outperforming existing methods by a significant

margin on the standard SQuAD benchmark.

1.4 Outline

This dissertation is structured as follows: in Chapter 2, we discuss related work on question answering, question generation and relevant methodologies. Then we present our novel neural network based KBQA model in Chapter 3, and the GNN based conversational machine reading comprehension model in Chapter 4. The graph2seq based approaches for question generation from KGs and question generation from text are introduced in Chapter 5 and Chapter 6, respectively. Finally, we conclude this dissertation and discuss potential future directions in Chapter 7.

CHAPTER 2
RELATED WORK

2.1 Knowledge Base Question Answering

With the rapid growth in large-scale knowledge bases (KBs) such as DBPedia [11], FreeBase [12] and Yago3 [13], knowledge base question answering (KBQA) has drawn increasing attention over the past few years. *Given questions in natural language (NL), the goal of KBQA is to automatically find answers from the underlying KB*, which provides a more natural and intuitive way to access the vast underlying knowledge resources.

The approaches proposed to tackle the KBQA task can be roughly categorized into two groups: semantic parsing (SP) and information retrieval (IR) approaches.

2.1.1 Semantic Parsing-based Approaches

SP-based approaches address the problem by constructing a semantic parser that converts natural language questions into intermediate logic forms, which can be further executed against the underlying KB. Traditional semantic parsers [124]–[126] require annotated logical forms as supervision, and are limited to narrow domains with a small number of logical predicates. Recent efforts overcome these limitations via the construction of hand-crafted rules or features [3], [127]–[134], schema matching [135], and using weak supervision from external resources [3], [132], [134], [136]–[138].

Various strategies have been explored to generate semantic query graphs from NL questions. [3] builds a coarse mapping from phrases to predicates using a KB and a large text corpus. A bridging operation is later used to generate additional predicates based on neighboring predicates. [128] searches partial logical forms via an agenda-based strategy that

Portions of this chapter previously appeared as: Y. Chen, L. Wu, and M. J. Zaki, "Bidirectional attentive memory networks for question answering over knowledge bases," in *Proc. 2019 Conf. N. Amer. Chap. Assoc. Comput. Ling.: Human Lang. Technol.*, vol. 1, Jun. 2-7, 2019, pp. 2913–2923.

Portions of this chapter previously appeared as: Y. Chen, L. Wu, and M. J. Zaki, "Graphflow: Exploiting conversation flow with graph neural networks for conversational machine comprehension," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Jul. 2020, pp. 1230–1236. Copyright © 2020, IJCAI (`https://www.ijcai.org`).

Portions of this chapter have been submitted to: Y. Chen, L. Wu, and M. J. Zaki, "Toward subgraph guided knowledge graph question generation with graph neural networks," in *Proc. 2020 Conf. Empirical Meth. Natural Lang. Process.*

Portions of this chapter previously appeared as: Y. Chen, L. Wu, and M. J. Zaki, "Reinforcement learning based graph-to-sequence model for natural question generation," in *Proc. 8th Int. Conf. Learn. Representations*, Apr. 26-30, 2020. [Online]. Available: https://openreview.net/forum?id=HygnDhEtvr

controls the order in which derivations are constructed. [134] formulates this query graph generation problem as a staged search problem. [129] convert a dependency tree into logic forms in three steps: binarization, substitution, and composition. To solve the ambiguity in the conversion process, [132] pushes down the disambiguation step into the query evaluation stage. Unlike many SP-based methods that rely on hand-crafted templates, [131] proposed a method for automatically generating templates that map a question into a triple pattern query over a KB. Notably, some SP-based approaches also exploit IR-based techniques [15], [130], [133], [134], [137], [138]. [137] proposes a trainable alignment model to directly map the natural language questions to KB relations. [133] proposes a learning-to-rank method to improve the entity and relation matching, while still using a number of hand-crafted templates and features. [15] maps natural language questions into logical forms via joint relational embeddings. A Siamese convolutional neural network is used in [134] and [130] to compute the similarity of two sequences (e.g., questions and relation paths), which is later used as a feature along with many other hand-crafted features in a learning-to-rank algorithm. [138] proposes a neural network-based answer type prediction model that improves the performance of existing semantic parsers. Unlike previous SP-based methods, [28] proposes a neural symbolic machine (NSM) for semantic parsing with weak supervision that does not require feature engineering or domain-specific knowledge, and is trained end-to-end via REINFORCE [111]. NSM contains a neural "programmer" that maps natural language questions to programs and a symbolic "computer" that executes programs against a KB and helps find good programs by pruning the search space. In general, most SP-based approaches more or less rely on a pre-defined set of rules or hand-crafted features, which limit their scalability and transferability.

2.1.2 Information Retrieval-based Approaches

Unlike SP-based approaches that usually assume a pre-defined set of lexical triggers or rules, which limit their domains and scalability, IR-based approaches directly retrieve answers from the KB in light of the information conveyed in the questions. They usually do not require hand-made rules and can therefore scale better to large KBs. Recently, deep neural networks have been shown to produce very strong results on many NLP tasks such as speech recognition [139], machine translation [140], and reading comprehension [141]. In the field of KBQA, under the umbrella of IR-based approaches, many embedding-based

methods [4], [5], [9], [10], [15]–[18], [142] have been proposed in the last few years and have shown promising results. These methods adopt various ways to encode questions and KB subgraphs into a common embedding space and directly match them in that space, and can be typically trained in an end-to-end manner.

[4] was the first to apply an embedding-based approach for KBQA that mapped questions and KB triples into the same embedding space, where a dot product is used to find the most relevant answer. On the other hand, [5] proposes the idea of subgraph embedding, which encodes more information (e.g., answer path and context) about the candidate answer. In follow-up work [10], [16], memory networks [143] are used to store candidate answers, and can be accessed iteratively to mimic multi-hop reasoning. Unlike the above methods that mainly use a bag-of-words (BOW) representation to encode questions and KB resources, [9], [18] apply more advanced network modules (e.g., CNNs and LSTMs) to encode questions. Hybrid methods have also been proposed [17], [142], which achieve improved results by leveraging additional knowledge sources such as Wikipedia free text. While most embedding-based approaches encode questions and answers independently, [18] proposes a cross-attention mechanism to encode questions according to various candidate answer aspects.

2.1.3 Complex Question Answering

In this section, we discuss the recent attempts on tackling the problem of complex question answering which typically requires an agent to perform a series of discrete symbolic operations in order to get the answers.

Considering the many advantages of deep learning approaches, one may hope they are highly effective for solving the problem of complex question answering. Researchers have proposed fully neuralized QA systems for performing complex reasoning, in other words, all of the discrete operations and reasoning results are represented by real-value vectors. [6] proposed to realizes the execution of compositional queries as a series of differentiable operations, with intermediate results saved in multiple layers of memory. However, they did not explicitly parameterize the discrete operations and thus only supported litmited operations (e.g., entry selection and aggregation). [26] presented the neural programmer-interpreter (NPI) which is a recurrent and compositional neural network augmented with a small set of basic arithmetic and logic operations that can be trained end-to-end with

weak supervision of question-answer pairs, and does not require domain-specific grammars, rules, or annotations which are key elements in previous approaches to program induction. The model runs for a fixed number of time steps using a recurrent neural network. At each step, it can select a segment in the data source and a particular operation to apply to that segment. [27] enhanced the NPI model with more built-in discrete operations and demonstrated its effectiveness on a real-world TBQA dataset. One great advantage of these fully neuralized approaches is that they embed everthing in a continuous vector space and the whole system can be trained via backpropagation. However, neural networks cannot support precise program execution and storing numerical intermediate results during the reasoning process as vectors does not seem reliable.

In order to overcome the above limitations of fully neuralized systems, researchers have explored many ways to combine neural networks and symbolic methods. Many of them try to map the natural language query into a program such as SQL which can be executed by a symbolic executor. A neural programmer is used to sample operations and associated arguments at each step and the intermediate results are returned by a symbolic executor by evaluating the partial program generated so far. [7] proposed a similar neural programmer. However, fully supervised execution traces of each program are required in order to train the model in an end-to-end manner. In order to train such a neural programmer-like model without intermediate supervision, reinforcement learning [111] is often adopted. [25] proposed a neural network for translating natural language questions to corresponding SQL queries. By leveraging the structure of SQL queries, the proposed Seq2SQL model significantly reduces the output space of generated queries. Moreover, they used rewards from in-the-loop query execution over the database to learn a policy to generate unordered parts of the query. [28] proposed a neural symbolic machine (NSM) for semantic parsing with weak supervision, which is trained end-to-end via reinforcement learning. While directly training a model from scratch using reinforcement learning is challenging, some approaches [25], [28] proposed to first pretrain the models in a fully or weakly supervised manner and then continue training using reinforcement learning. [144] developed a probabilistic deductive database, TensorLog, where inference tasks can be compiled into sequences of differentiable operations. Inspired by TensorLog, [30] proposed the Neural Logic Programming framework, that combines the parameter and structure learning of first-order logical rules in an end-to-end differentiable model.

2.2 Machine Reading Comprehension

2.2.1 Traditional Machine Reading Comprehension

The goal of machine reading comprehension (MRC) is to answer a natural language question using the given passage. Recently, impressive progress has been made in the MRC task [8], [141], [145]–[148], with most methods relying on (co-)attention mechanisms that capture the interaction between the question and its context. [145] proposed an attention based recurrent neural network framework which consists of two novel components: an attentive reader which focuses on the passages of a context document which are most likely to answer the query and an impatient reader which rereads from the document as each query token is read. [146] presented a simple but effective model called attention-over-attention reader which induces "attended attention" for answer predictions by placing another attention mechanism over the document-level attention. [8] proposed a multi-stage hierarchical process that represents the context at different levels of granularity and uses a bi-directional attention flow mechanism to achieve a query-aware context representation without early summarization. [147] proposed the Dynamic Coattention Network which fuses co-dependent representations of the question and the document in order to focus on relevant parts of both. [141] developed an extraction-then-synthesis framework to synthesize answers from extraction results which could be able to handle the cases where the answer is not an exact text span in a passage. [148] proposed a multi-factor attentive encoding that aggregates meaningful facts even when they are located in multiple sentences.

2.2.2 Conversational Machine Reading Comprehension

Recent years have observed a surge of interest in conversational machine reading comprehension (MRC). Unlike the traditional setting of MRC that requires answering a single question given a passage (aka context), the conversational MRC task is to answer the current question in a conversation given a passage and the previous questions and answers. The goal of this task is to mimic real-world situations where humans seek information in a conversational manner.

Despite the success existing works have achieved on traditional MRC (e.g., SQuAD [149]), conversational MRC has proven significantly more challenging when the conversations are incorporated into the MRC task. It has been observed that in those human-to-human conversations [37], [38], the starting turns tend to focus on the beginning chunks of the

passage and as the conversation progresses, the focus shifts to the later chunks. Moreover, the turn transitions are usually smooth, with the focus often remaining in the same chunk or moving to a neighboring chunk. Lastly, many questions refer back to the conversation history via either coreference or ellipsis. Therefore, one hopes to develop a model that can capture these shifts of focus during a conversation. We model *conversation flow* as a sequence of latent states in the dialog and learn important latent states associated with these shifts of focus.

To cope with the above challenges, many methods have been proposed to effectively utilize conversation history, including previous questions and/or previous answers. Most existing approaches, however, simply prepend the conversation history to the current question [37], [150] or add previous answer locations to the passage [38], [43], and treat the task as a single-turn MRC while ignoring the important information from the conversation flow. [151] assumed that the hidden representations generated during the previous reasoning processes potentially capture important information for answering the previous questions, and thus provide additional clues for answering the current question. They proposed an *Integration-Flow* (IF) mechanism to first sequentially process the passages, in parallel to the question turns, and then to sequentially process the question turns, in parallel to passage words. Their FlowQA model achieves strong empirical results on two benchmarks (i.e., CoQA and QuAC) [37], [38].

However, the IF mechanism is not quite natural since it does not mimic how humans perform reasoning. This is because when humans execute such task, they typically do not first perform reasoning in parallel for each question, and then refine the reasoning results across different turns. This may partially explain why this strategy is inefficient because the results of previous reasoning processes are not incorporated into the current reasoning process. As a result, the reasoning performance at each turn is only slightly improved by the hidden states of the previous reasoning process, even though they use stacked IF layers to try to address this problem.

2.3 Natural Question Generation from Knowledge Graphs

Recent years have seen a surge of interests in Question Generation (QG) in machine learning and natural language processing. The goal of QG is to generate a natural language (NL) question for a given form of data such as text [73], [75], [152], [153], images [53]–[58],

tables [59]–[62], and knowledge graphs (KGs) [63]–[69].

KGs have drawn a large amount of research attention in recent years, partially due to their huge potential for an accessible, natural way of retrieving information without a need for learning complex query languages such as SPARQL. In order to train a large KB question answering (QA) system, a large number of QA pairs are often needed, which can be a severe bottleneck in practice. Developing effective approaches to generate high-quality QA pairs from KGs can significantly address the data scarcity issue for KBQA.

Early works [63]–[65] on QG from KGs mainly focused on template-based approaches that require significant amount of human effort, and have low generalizability and scalability. Recently, Seq2Seq [77], [78] based neural architectures have been applied to this task without resort to manually-designed templates and are end-to-end trainable. However, these methods [66]–[68] only focus on generating simple questions from a single triple as they typically employ an RNN-based encoder which cannot handle graph-structured data. Very recently, [69] presented a Transformer [108] based encoder-decoder model that allows to encode a KG subgraph and generate multi-hop questions. This is probably the first NN-based method that deals with QG from a KG subgraph instead of just a single triple. However, their method treats a KG subgraph as a set of triples, which does not distinguish between entities and relations while modeling the graph, and also does not utilize explicit connections among triples.

In summary, existing approaches for QG from KGs suffer from several limitations; they i) mostly focus on a simple setting which is to generate questions from a single KG triple, and ii) they build on either RNN-based or Transformer-based models to encode a linearized KG sugraph, which totally discards the explicit structure information of a KG subgraph.

2.4 Natural Question Generation from Text

Conventional methods [70]–[72] for QG from text rely on heuristic rules or hand-crafted templates, leading to the issues of low generalizability and scalability. Recent attempts have focused on NN-based approaches that do not require manually-designed rules and are end-to-end trainable. Specifically, attention-based Seq2Seq models [79], [140] and their enhanced versions with copy [154], [155] and coverage [156] mechanisms have been widely applied and show promising results on this task [73]–[76]. However, these methods typically ignore the hidden structural information associated with a word sequence such as the syntactic parsing

tree. Failing to utilize the rich text structure information beyond the simple word sequence may limit the effectiveness of these models for QG.

In addition, various ways [74], [87], [89] have been proposed to utilize the target answer for guiding the question generation. [74] marked answer positions when encoding the input passage. [75] proposed a multi-perspective matching strategy to match the answer with the passage. [90] proposed an answer-separated Seq2Seq model which treats the passage and the answer separately. [88] proposed a multi-task learning framework to guide the model to learn the accurate boundaries between copying and generation. However, these methods neglect potential semantic relations between passage words and answer words, and thus fail to explicitly model the global interactions among them in the embedding space.

It has been observed that in general, cross-entropy based sequence training has several limitations like exposure bias and inconsistency between train/test measurement [109], [110]. To address the limitations of cross-entropy based sequence learning, some recent approaches [84], [89] aim at directly optimizing evaluation metrics using REINFORCE. [84], [89] augmented an RNN-based generator with an evaluator which evaluates and assigns a reward to each predicted question. Concurrent works have explored tackling the QG task with various semantics-enhanced rewards [101] or large-scale pretrained language models [157].

In summary, existing approaches for QG from text suffer from several limitations; they i) ignore the rich structure information hidden in text, ii) solely rely on cross-entropy loss that leads to issues like exposure bias and inconsistency between train/test measurement, and iii) fail to fully exploit the answer information.

2.5 Modern Deep Learning and Reinforcement Learning Methods

Deep Learning (DL) [158], [159] is a class of machine learning algorithms that use multi-layer neural networks to progressively extract higher level features from the raw input in an end-to-end manner. These methods have dramatically improved the state-of-the-art in CV [160], speech recognition [161], NLP [123], [162] and many other domains such as healthcare [99], drug discovery [163], finance [164], genomics [165], and software [166]. Reinforcement learning (RL) [167], [168] is an area of machine learning concerned with how agents should take actions in an environment in order to maximize the cumulative reward. Recent years have seen very successful applications of RL in games, robotics, computer vision, NLP and many other domains such as healthcare. In this section, we will introduce

some relevant background knowledge on deep learning and reinforcement learning.

2.5.1 Recurrent Neural Networks

Recurrent neural networks (RNN) [169] is a classical family of neural networks where connections between nodes form a directed graph to exploit temporal information in sequential data. Basically, RNNs perform the same task for every element of a sequence, with the output being dependent on the computational results at the previous time step. RNNs are characterized by their internal memory, or hidden layer, defined by the recurrence relation:

$$\mathbf{h}_t = f_h(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{xh}\mathbf{x}_t + \mathbf{b}_h) \tag{2.1}$$

where $\mathbf{W}_{hh}$ and $\mathbf{W}_{xh}$ are both weight matrices, and $\mathbf{b}_h$ is a weight vector. $\mathbf{x}_t$ and $\mathbf{h}_t$ are the input vector and the hidden state at the t-th time step, respectively. $f_h(\cdot)$ is an element-wise non-linear activation function, and $\mathbf{h}^{(0)}$ is an additional model parameter indicating the initial hidden state. The output vector can be further computed by

$$\mathbf{y}_t = f_y(\mathbf{W}_y\mathbf{h}_t + \mathbf{b}_y) \tag{2.2}$$

where $\mathbf{W}_y$ is a weight matrix, and $\mathbf{b}_y$ is a weight vector. $\mathbf{y}_t$ is the output vector at the t-th time step, and $f_y(\cdot)$ is an element-wise non-linear activation function.

RNNs incorporate an internal memory $\mathbf{h}_t$ that can, in principle, summarize the entire sequence history, which makes them well suited to capture long-term dependencies. However, studies have showed that it is challenging to train RNNs efficiently by gradient-based optimization because of the the gradient vanishing/explosion problem for long sequences of inputs [170], [171].

Long Short-Term Memory (LSTM) [172] network is a type of RNN which is good at capturing long-range context dependences over input sequences. Similar to regular RNNs, LSTM uses a sequence of memory cells to store and memorize historical information. The major difference is that each memory cell of LSTM contains three gates (input gate, forget gate, and output gate) to control the information flow from the previous time step to the current time step. The introduced gating mechanism enables LSTM to alleviate the gradient vanishing/explosion problem for long sequences of inputs.

Given an input sequence of vectors $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_T)$, an LSTM network outputs a

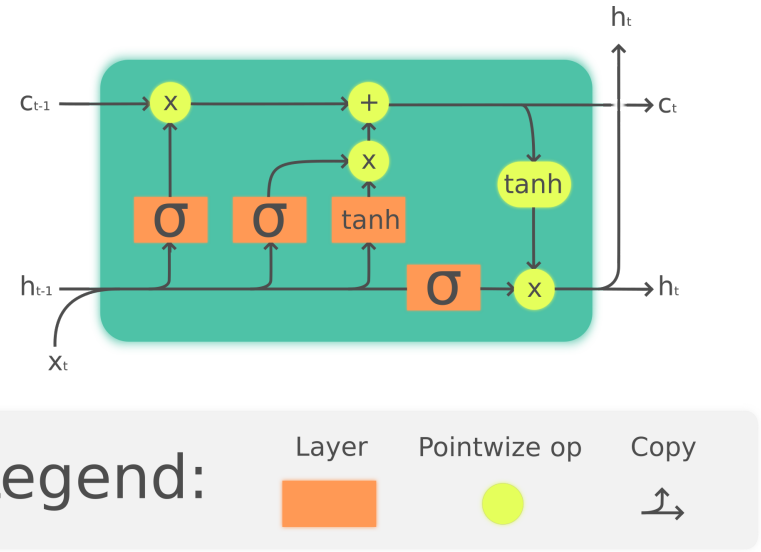

Fig. 2.1: Architecture of the Long Short-Term Memory (LSTM) network. The picture is from `https://commons.wikimedia.org/wiki/File:The_LSTM_cell.png`, licensed under version 4.0 of the Creative Commons CC-BY license (CC BY 4.0).

sequence of hidden states $\mathbf{H} = (\mathbf{h}_1, ..., \mathbf{h}_T)$. A memory cell at the t-th time step digests the input vector $\mathbf{x}_t$ and the previous hidden state $\mathbf{h}_{t-1}$ to produce the current hidden state $\mathbf{h}_t$ as follows:

$$\mathbf{f}_t = \sigma(\mathbf{W}_f[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f) \tag{2.3a}$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_i[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i) \tag{2.3b}$$

$$\tilde{\mathbf{C}}_t = \tanh(\mathbf{W}_C[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_C) \tag{2.3c}$$

$$\mathbf{C}_t = \mathbf{f}_t \odot \mathbf{C}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{C}}_t \tag{2.3d}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o) \tag{2.3e}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{C}_t) \tag{2.3f}$$

where $\mathbf{W}$ terms are weight matrices, $\mathbf{b}$ terms are bias vectors, and σ and tanh are non-linear activation functions. $\mathbf{C}_t$ and $\mathbf{h}_t$ are the cell state and hidden state at the current time step, respectively. $\mathbf{f}_t$, $\mathbf{i}_t$ and $\mathbf{o}_t$ are forget gate (i.e., deciding how much information to forget from the previous cell state $\mathbf{C}_{t-1}$), input gate (i.e., deciding how much information to keep from the new candidate vector $\tilde{\mathbf{C}}_t$) and output gate (i.e., deciding how much information to

output), respectively.

Due to the huge success of LSTM networks on modeling sequential data [77], [161], many RNN variants [78], [173]–[175] have been proposed with the same spirit of the gating mechanism used in LSTM. Besides LSTM, another popular RNN variant is the Gated Recurrent Unit (GRU) [78]. GRU makes two major changes to LSTM in order to simplify LSTM while reserving its expressive power. Compared to LSTM, GRU combines the forget and input gates into a single "update gate", and merges the cell state and hidden state. The resulting model is simpler than standard LSTM models, and has achieved comparative or even better results on various tasks. A few comparative studies [176], [177] have been conducted to systematically compare different RNN variants across various tasks.

A natural extension to RNN/LSTM is to model the sequence from both positive time direction and negative time direction, which is proposed as bidirectional RNN [178].

RNNs have many applications in sequence modeling tasks, such as speech recognition [161], [179], [180], hand writing recognition [181], machine translation [77], [78], question generation [73], [75], text summarization [41], [80], [81] and dialog systems [82], [83].

2.5.2 Attention Mechanisms

Attention mechanisms help neural networks focus on important parts of the features. The idea of attention mechanisms was first introduced in the literature of machine translation by [140]. Before attention mechanisms, neural machine translation approaches typically encoded a complete sentence into a fixed-length vector, and generated the target sentence solely based on the compressed information. As one can imagine, a sentence with hundreds of words represented by a fixed-length vector will surely lead to information loss and thus inadequate translation. Attention mechanisms partially fix this issue by allowing the machine translator to look over all the information contained in the source sentence, and then at decoding time to generate the next word according to the current word and the context. The context is essentially the weighted sum of word vectors in the source sentence where the weights come from an attention mechanism. Thereafter, attention mechanisms have found broad application in all kinds of NLP tasks including question answering [8], [18], [97], [145]–[147], [182]–[184].

Many variants of attention mechanisms have been proposed. Here we just introduce some widely used variants.

Additive attention

$$e_i = \mathbf{v}^T \tanh(\mathbf{W}[\mathbf{x}, \mathbf{h}_i]) \tag{2.4}$$

where $\mathbf{W}$ and $\mathbf{v}$ are $d \times 2d$ and d-dimensional trainable weights, respectively.

Multiplicative attention

$$e_i = \mathbf{x}\mathbf{W}\mathbf{h}_i \tag{2.5}$$

where $\mathbf{W}$ is a $d \times d$ trainable weight matrix.

Multiplicative attention version 2

$$e_i = \text{ReLU}(\mathbf{W}\mathbf{x})^T \text{ReLU}(\mathbf{W}\mathbf{h}_i) \tag{2.6}$$

where $\mathbf{W}$ is a $d \times d$ trainable weight matrix.

2.5.3 Memory Networks

Memory networks are neural networks equipped with a long-term memory component that can be read and written to. It was first introduced by [143] in the context of question answering where the long-term memory acts as a dynamic knowledge base. In memory networks, the input and response languages as well as the storage languages (here, the facts from KBs) are embedded in the same vector space. The memory networks maintain an internal state vector which can be used to access the memory component typically via an attention mechanism, and the information read from the memory component can be used to update the internal state vector. This kind of read-and-update operation is called one-hop reasoning. Memory networks can benefit from performing multi-hop reasoning in practice. [185] extended the basic memory networks to key-value memory networks as shown in Fig. 2.2. Unlike a basic memory network, in a key-value memory network, the addressing stage is based on the key memory while the reading stage uses the value memory, which gives greater flexibility to encode prior knowledge via functionality separation. Memory networks have shown promising results in KBQA [6], [10], [16], [186], reading comprehension [185] and dialog systems [187].

Notably, most neural programmer-like models we discussed in Section 2.1.3 adopt the architecture of memory networks. They often equip a learnable memory component which

stores the vector representations of predefined symbolic operations. The operation sampling process is done by reading relevant memory slots from the memory bank into a controller via some attention mechanism.

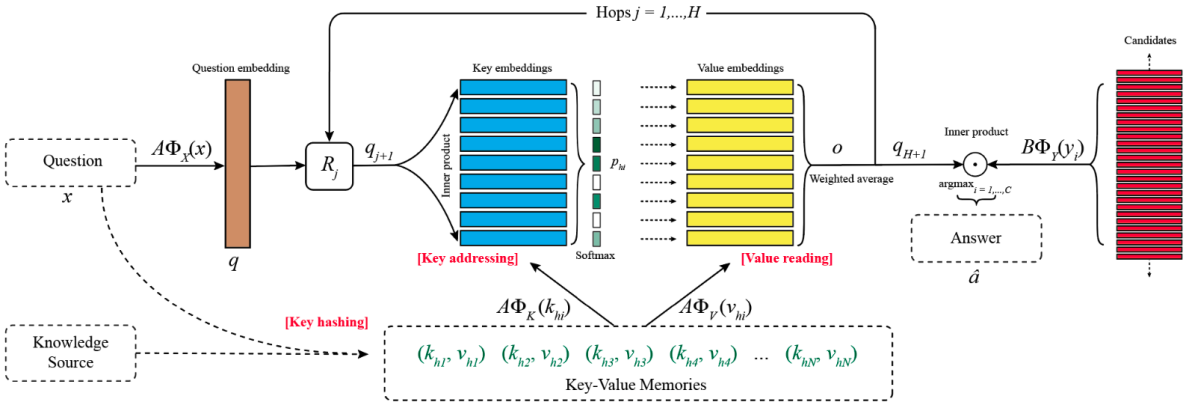

Fig. 2.2: A Key-Value Memory Network for question answering. The picture is from [185], licensed under version 4.0 of the Creative Commons CC-BY license (CC BY 4.0).

2.5.4 Graph Neural Networks

Graph embedding approaches aim at learning meaningful graph node embeddings and/or graph embeddings which preserve network properties. Previous graph embedding approaches such as DeepWalk [188], node2vec [189], LINE [190], TADW [191] share no parameters between nodes in the encoder, hence the number of parameters grows linearly with the number of nodes. And those direct embedding methods lack the ability of generalization, which means they cannot deal with dynamic graphs or generalize to new graphs.

Over the past few years, graph neural networks (GNNs) [192]–[199] have drawn increasing attention since they extend traditional Deep Learning approaches to non-euclidean data such as graph-structured data. [192] proposed the first graph neural network model that extends existing neural network methods for processing the data represented in the graph domain. Early works [200]–[203] on GNNs aimed to generalize the convolution operation in the Fourier domain to the graph domain via spectral analysis. [193] proposed the Graph Convolutional Network (GCN) which simplifies spectral convolutions on graphs by restricting the filters to operate in a 1-hop neighborhood around each node. [197] proposed the Gated Graph Sequence Neural Networks (GGSNN) by employing the Gated Recurrent Unit (GRU) [78] in GNNs. [196] introduced GraphSAGE, a general inductive framework

that allows node embeddings to be efficiently generated for unseen nodes. [204] applied the idea of multi-head attention [108] to GNNs and proposed the Graph Attention Network (GAT) which can learn different weights to different neighboring nodes when performing node aggregation. [205] proposed Gated Attention Networks (GaAN) which enhances the GAT by using a convolutional sub-network to control the importance of different attention heads. [199] presented a unified GNN framework which decomposes the GNN computation into the edge update stage, the node update stage and the global update stage. As shown in Fig. 2.3, given the node feature vectors and the adjacency matrix, GNNs can update all the node embeddings in parallel by incorporating information from neighboring nodes and/or edges. Once the node embeddings are learned, graph-level embeddings can be obtained by applying graph pooling techniques to the node embeddings such as average pooling, max pooling, DiffPool [206] and gPool [207].

While the most straightforward applications of GNNs are tasks such as node classification, link prediction and graph classification, considering the fact that GNNs are good at modeling relations among elements, they have many successful applications in computer vision [208]–[211], recommender systems [212]–[216], and drug discovery [163], [217]–[222]. GNNs also have broad applications in NLP such as text classification [223], KBQA [224], MRC [225]–[227], dialog systems [228], [229], machine translation [230], [231], semantic parsing [232], and graph-to-text generation (e.g., AMR, SQL and KG to text) [198], [227], [233]–[237].

For tasks where the graph structure is unknown, [225], [226], [232] construct a static graph where entity mentions in a passage are nodes of this graph and edge information is determined by coreferences of entity mentions. [209] dynamically construct a graph which contains all the visual objects appearing in an image. [238], [239] dynamically constructs a graph of all words from free text. [240] proposed the LDS model for jointly learning the graph and the parameters of GNNs by leveraging the bilevel optimization technique. [239] proposed an iterative deep graph learning (IDGL) framework for jointly and iteratively learning the graph structure and graph embedding that are optimized toward the downstream prediction tasks.

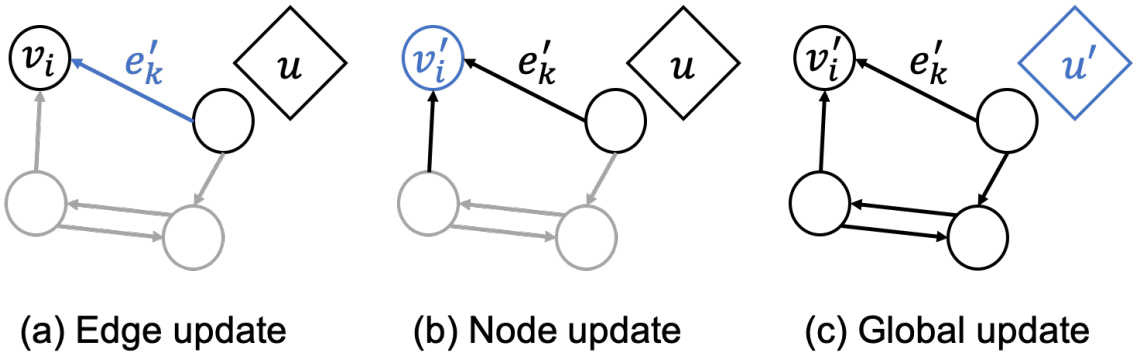

Fig. 2.3: Computation steps in a graph neural network block.

2.5.5 Reinforcement Learning

Reinforcement learning (RL) [167], [168] is an area of machine learning concerned with how agents should take actions in an environment in order to maximize the notion of cumulative reward. RL differs from supervised learning in requiring no labeled input/output pairs be presented, and no sub-optimal actions to be explicitly corrected.

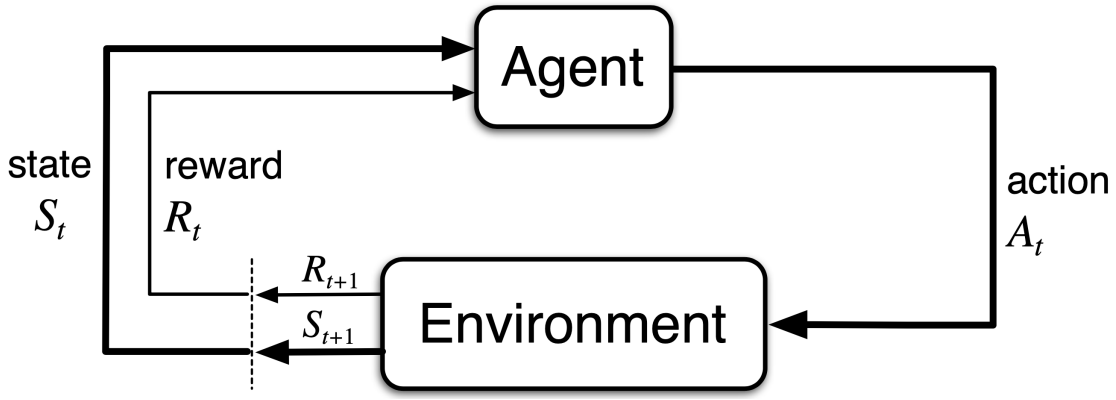

Fig. 2.4: The agent–environment interaction in reinforcement learning. The picture is from [168], licensed under Attribution-NonCommercial-NoDerivs 2.0 Generic license (CC BY-NC-ND 2.0).

Basic RL is modeled as a Markov decision process [241]:

- a set of environment and agent states, S;
- a set of actions (A) of the agents;
- $P_a(s, s') = Pr(s_{t+1} = s' | s_t = s, a_t = a)$ is the probability of transition (at time t) from state s to state s' under action a;

- $R_a(s, s')$ is the immediate reward after transition from s to s' with action a;
- rules that describe what the agent observes.

A reinforcement learning agent interacts with its environment in discrete time steps. At each time t, based on the observation o_t and the associated reward r_t, the agent chooses an action a_t from the set of available actions, which is subsequently sent to the environment. In order to build an optimal policy, the agent faces the dilemma of exploring new states while maximizing its reward at the same time. This is called Exploration vs Exploitation trade-off [242]. Deep reinforcement learning [243], [244] extends reinforcement learning by using a deep neural network and without explicitly designing the state space. Researchers have developed many model-free RL algorithms such as Q-learning [245], SARSA (State-Action-Reward-State-Action) [246], DQN [247] and DDPG (Deep Deterministic Policy Gradient) [248]. RL has been widely applied in many fields such as games [247], [249]–[253], robotics [254]–[256], NLP [257]–[259], computer vision [260]–[262] and healthcare [263]–[267].

CHAPTER 3
KNOWLEDGE BASE QUESTION ANSWERING

3.1 Overview

When answering natural language questions over knowledge bases (KB), different question components and KB aspects play different roles. However, most existing embedding-based methods for knowledge base question answering (KBQA) ignore the subtle interrelationships between the question and the KB (e.g., entity types, relation paths and context). In this work, we propose to directly model the two-way flow of interactions between the questions and the underlying KB via a novel **B**idirectional **A**ttentive **M**emory **net**work, called **BAMnet**. We assume that the world knowledge (i.e., the KB) is helpful for better understanding the questions. Similarly, the questions themselves can help us focus on important KB aspects. To this end, we design a *two-layered bidirectional attention network*. The *primary attention network* is intended to focus on important parts of a question in light of the KB and important KB aspects in light of the question. Built on top of that, the *secondary attention network* is intended to enhance the question and KB representations by further exploiting the two-way attentions. Specifically, we start by computing a question summary using a self-attention mechanism that captures its semantic meaning without considering the KB. Based on the question summary, a KB summary is then computed, which summarizes the KB knowledge relevant to answering the question. Given the KB summary, we are able to pay different levels of attention to different parts of the question, yielding a KB-aware representation for the question. Similarly, we learn a question-aware representation for the KB. Later, the representations of the question and KB are further enhanced to better capture the interactions between them. Through this idea of *hierarchical two-way attentions*, we are able to distill the information that is the most relevant to answering the questions on both sides of the question and KB.

We highlight the contributions of this work as follows:

- We propose a novel bidirectional attentive memory network which is intended to directly model the two-way interactions between questions and the KB for the task of

This chapter previously appeared as: Y. Chen, L. Wu, and M. J. Zaki, "Bidirectional attentive memory networks for question answering over knowledge bases," in *Proc. 2019 Conf. N. Amer. Chap. Assoc. Comput. Ling.: Human Lang. Technol.*, vol. 1, Jun. 2-7, 2019, pp. 2913–2923.

KBQA.

- On the WebQuestions benchmark, our method significantly outperforms previous IR-based methods while remaining competitive with (hand-crafted) SP-based methods.
- By design, our method offers good interpretability thanks to the attention mechanisms.

3.2 Approach: BAMnet

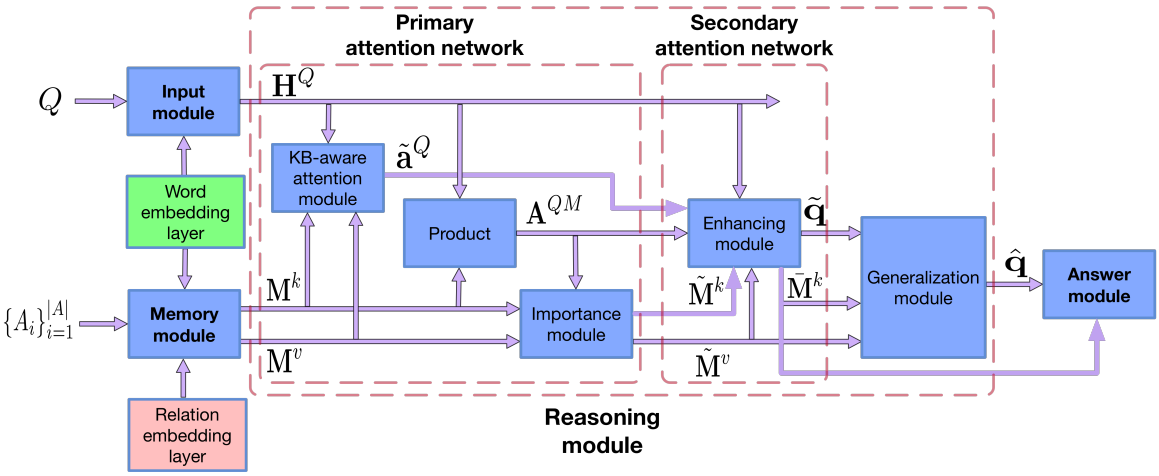

Fig. 3.1: Overall architecture of the BAMnet model.

Given an NL question, the goal is to fetch answers from the underlying KB. Our proposed BAMnet model consists of four components which are the *input module*, *memory module*, *reasoning module* and *answer module*, as shown in Fig. 3.1.

3.2.1 Input Module

An input NL question $Q = \{q_i\}_{i=1}^{|Q|}$ is represented as a sequence of word embeddings (q_i) by applying a word embedding layer. We then use a bidirectional LSTM [172] to encode the question as $\mathbf{H}^Q$ (in $\mathbb{R}^{d \times |Q|}$) which is the sequence of hidden states (i.e., the concatenation of forward and backward hidden states) generated by the BiLSTM.

3.2.2 Memory Module

Candidate generation. Even though all the entities from the KB could in principle be candidate answers, this is computationally expensive and unnecessary in practice. We only consider those entities which are "close" to the main topic entity of a question. An answer is

the text description (e.g., a name) of an entity node. For example, *Ohio* is the topic entity of the question "Who was the secretary of state of Ohio in 2011?" (see Fig. 3.2). After getting the topic entity, we collect all the entities connected to it within h hops as candidate answers, which we denote as $\{A_i\}_{i=1}^{|A|}$.

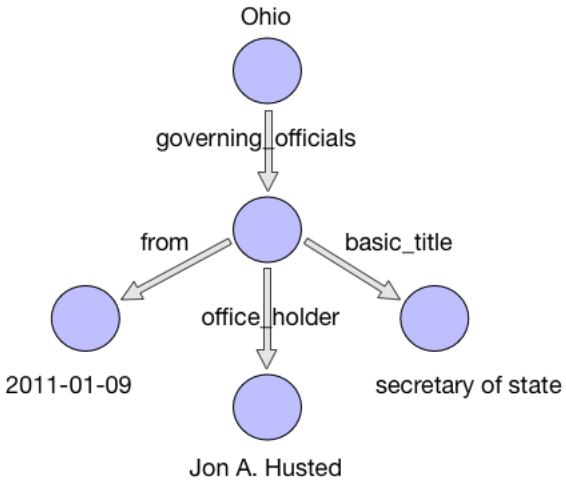

Fig. 3.2: A working example from Freebase. Relations in Freebase have hierarchies where high-level ones provide too broad or even noisy information about the relation. Thus, we choose to use the lowest level one.

KB representation. For each candidate answer from the KB, we encode three types of information: answer type, path and context.

Answer type Entity type information is an important clue in ranking answers. For example, if a question uses the interrogative word *where*, then candidate answers with types relevant to the concept of location are more likely to be correct. We use a BiLSTM to encode its text description to get a d-dimensional vector $\mathbf{H}_i^{t_1}$ (i.e., the concatenation of last forward and backward hidden states).

Answer path. We define an answer path as a sequence of relations from a candidate answer to a topic entity. For example, for the *Ohio* question (see Fig. 3.2), the answer path of *Jon A. Husted* can be either represented as a sequence of relation ids [*office_holder, governing_officials*] or the text description [*office, holder, governing, officials*]. We thus encode an answer path as $\mathbf{H}_i^{p_1}$ via a BiLSTM, and as $\mathbf{H}_i^{p_2}$ by computing the average of its relation embeddings via a relation embedding layer.

Answer context. The answer context is defined as the surrounding entities (e.g., sib-

ling nodes) of a candidate which can help answer questions with constraints. For example, in Fig. 3.2, the answer context of *Jon A. Husted* includes the government position title *secretary of state* and starting date *2011-01-09*. However, for simple questions without constraints, the answer context is unnecessary and can potentially incorporate noise. We tackle this issue with two strategies: 1) we use a novel *importance module* (explained later) to focus on important answer aspects, and 2) we only consider those context nodes that have overlap with the question. Specifically, for each context node (i.e., a sequence of words) of a candidate, we first compute the longest common subsequence between it and the question, we then encode it via a BiLSTM only if we get a non-stopwords substring. Finally, the answer context of a candidate answer will be encoded as the average of all context node representations, which we denote as $\mathbf{H}_i^c$.

Key-value memory module. In our model, we use a key-value memory network [185] to store candidate answers. Unlike a basic memory network [143], its addressing stage is based on the key memory while the reading stage uses the value memory, which gives greater flexibility to encode prior knowledge via functionality separation. Thus, after encoding the answer type, path and context, we apply linear projections on them as follows:

$$\mathbf{M}_i^{k_t} = f_t^k(\mathbf{H}_i^{t_1}) \qquad \mathbf{M}_i^{v_t} = f_t^v(\mathbf{H}_i^{t_1}) \tag{3.1a}$$

$$\mathbf{M}_i^{k_p} = f_p^k([\mathbf{H}_i^{p_1}; \mathbf{H}_i^{p_2}]) \qquad \mathbf{M}_i^{v_p} = f_p^v([\mathbf{H}_i^{p_1}; \mathbf{H}_i^{p_2}]) \tag{3.1b}$$

$$\mathbf{M}_i^{k_c} = f_c^k(\mathbf{H}_i^c) \qquad \mathbf{M}_i^{v_c} = f_c^v(\mathbf{H}_i^c) \tag{3.1c}$$

where $\mathbf{M}_i^{k_t}$ and $\mathbf{M}_i^{v_t}$ are d-dimensional key and value representations of answer type A_i^t, respectively. Similarly, we have key and value representations for answer path and answer context. We denote $\mathbf{M}$ as a key-value memory whose row $\mathbf{M}_i = \{\mathbf{M}_i^k, \mathbf{M}_i^v\}$ (both in $\mathbb{R}^{d\times 3}$), where $\mathbf{M}_i^k = [\mathbf{M}_i^{k_t}; \mathbf{M}_i^{k_p}; \mathbf{M}_i^{k_c}]$ comprises the keys, and $\mathbf{M}_i^v = [\mathbf{M}_i^{v_t}; \mathbf{M}_i^{v_p}; \mathbf{M}_i^{v_c}]$ comprises the values. Here $[,]$ and $[;]$ denote row-wise and column-wise concatenations, respectively.

3.2.3 Reasoning Module

The reasoning module consists of a generalization module, and our novel *two-layered bidirectional attention network* which aims at capturing the two-way interactions between questions and the KB. The *primary attention network* contains the KB-aware attention module which focuses on the important parts of a question in light of the KB, and the

importance module which focuses on the important KB aspects in light of the question. The *secondary attention network* (*enhancing module* in Fig. 3.1) is intended to enhance the question and KB vectors by further exploiting the two-way attention.

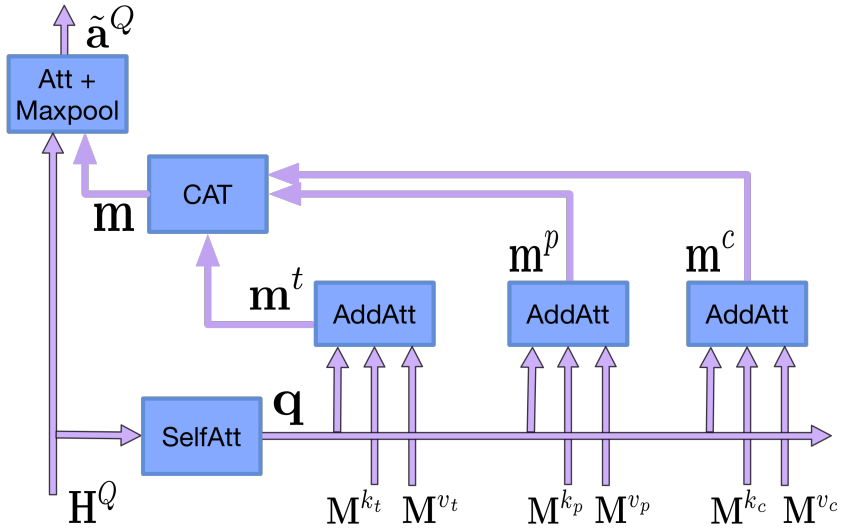

Fig. 3.3: KB-aware attention module. CAT: concatenation, SelfAtt: self-attention, AddAtt: additive attention.

KB-aware attention module. Not all words in a question are created equal. We use a KB-aware attention mechanism to focus on important components of a question, as shown in Fig. 3.3. Specifically, we first apply self-attention (SelfAtt) over all question word vectors $\mathbf{H}^Q$ to get a d-dimensional question vector $\mathbf{q}$ as follows

$$\mathbf{q} = BiLSTM([\mathbf{H}^Q\mathbf{A}^{QQ^T}, \mathbf{H}^Q]) \tag{3.2a}$$

$$\mathbf{A}^{QQ} = softmax((\mathbf{H}^Q)^T\mathbf{H}^Q) \tag{3.2b}$$

where *softmax* is applied over the last dimension of an input tensor by default. Here $\mathbf{A}^{QQ}$ computes the attention strengths among all words in a question, and concatenating $\mathbf{H}^Q$ and $\mathbf{H}^Q\mathbf{A}^{QQ^T}$ fuses word information with information from surrounding words, which is later fed into a BiLSTM to get a self-attentive question representation. Using question summary $\mathbf{q}$, we apply another attention (AddAtt) over the memory to obtain answer type $\mathbf{m}_t$, path

$\mathbf{m}_p$ and context summary $\mathbf{m}_c$:

$$\mathbf{m}_x = \sum_{i=1}^{|A|} a_i^x \cdot \mathbf{M}_i^{v_x} \tag{3.3a}$$

$$\mathbf{a}^x = \mathrm{Att}_{\mathrm{add}}(\mathbf{q}, \mathbf{M}^{k_x}) \tag{3.3b}$$

where $x \in \{t, p, c\}$, and $\mathrm{Att}_{\mathrm{add}}(\mathbf{x}, \mathbf{y}) = \mathit{softmax}(\tanh([\mathbf{x}^T, \mathbf{y}]\mathbf{W_1})\mathbf{W_2})$, with $\mathbf{W_1} \in \mathbb{R}^{2d \times d}$ and $\mathbf{W_2} \in \mathbb{R}^{d \times 1}$ being trainable weights.

So far, we have obtained the KB summary $\mathbf{m} = [\mathbf{m}_t; \mathbf{m}_p; \mathbf{m}_c]$ in light of the question. We proceed to compute the question-to-KB attention between question word q_i and KB aspects as formulated by $\mathbf{A}^{Qm} = \mathbf{H}^{Q^T}\mathbf{m}$. By applying max pooling over the last dimension (i.e., the KB aspect dimension) of $\mathbf{A}^{Qm}$, that is, $\mathbf{a}_i^Q = \max_j \mathbf{A}_{ij}^{Qm}$, we select the strongest connection between q_i and the KB. The idea behind it is that each word in a question serves a specific purpose (i.e., indicating answer type, path or context), and max pooling can help find out that purpose. We then apply a softmax over the resulting vector to obtain $\tilde{\mathbf{a}}^Q$ which is a *KB-aware question attention vector* since it indicates the importance of q_i in light of the KB.

Importance module. The importance module focuses on important KB aspects as measured by their relevance to the questions. We start by computing a $|Q| \times |A| \times 3$ attention tensor $\mathbf{A}^{QM}$ which indicates the strength of connection between each pair of $\{q_i, A_j^x\}^{x=\{t,p,c\}}$. Then, we take the max of the question word dimension of $\mathbf{A}^{QM}$ and normalize it to get an attention matrix $\tilde{\mathbf{A}}^M$, which indicates the importance of each answer aspect for each candidate answer. After that, we proceed to compute question-aware memory representations $\tilde{\mathbf{M}}^k$. Thus, we have:

$$\tilde{\mathbf{M}}^v = \{\tilde{\mathbf{M}}_i^v\}_{i=1}^{|A|} \in \mathbb{R}^{|A| \times d} \qquad \tilde{\mathbf{M}}_i^v = \sum_{j=1}^{3} \mathbf{M}_{ij}^v \tag{3.4a}$$

$$\tilde{\mathbf{M}}^k = \{\tilde{\mathbf{M}}_i^k\}_{i=1}^{|A|} \in \mathbb{R}^{|A| \times d} \qquad \tilde{\mathbf{M}}_i^k = \tilde{\mathbf{A}}_i^M \mathbf{M}_i^k \tag{3.4b}$$

$$\tilde{\mathbf{A}}^M = \mathit{softmax}(\mathbf{A}^{M^T})^T \qquad \mathbf{A}^M = \max_i \{\mathbf{A}_i^{QM}\}_{i=1}^{|Q|} \qquad \mathbf{A}^{QM} = \left(\mathbf{M}^k \mathbf{H}^Q\right)^T \tag{3.4c}$$

Enhancing module. We further enhance the question and KB representations by exploiting two-way attention. We compute the KB-enhanced question representation $\tilde{\mathbf{q}}$ which incorpo-

rates the relevant KB information by applying max pooling over the last dimension (i.e., the answer aspect dimension) of $\mathbf{A}^{QM}$, that is, $\mathbf{A}_M^Q = \max_k\{\mathbf{A}_{:,:,k}^{QM}\}_{k=1}^3$, and then normalizing it to get a question-to-KB attention matrix $\tilde{\mathbf{A}}_M^Q$ from which we compute the question-aware KB summary and incorporate it into the question representation $\tilde{\mathbf{H}}^Q = \mathbf{H}^Q + \tilde{\mathbf{a}}^Q \odot (\tilde{\mathbf{A}}_M^Q \tilde{\mathbf{M}}^v)^T$. Finally, we obtain a d-dimensional KB-enhanced question representation $\tilde{\mathbf{q}} = \tilde{\mathbf{H}}^Q \tilde{\mathbf{a}}^Q$.

Similarly, we compute a question-enhanced KB representation $\overline{\mathbf{M}}^k$ which incorporates the relevant question information:

$$\overline{\mathbf{M}}^k = \tilde{\mathbf{M}}^k + \tilde{\mathbf{a}}^M \odot (\tilde{\mathbf{A}}_Q^M (\tilde{\mathbf{H}}^Q)^T) \tag{3.5a}$$

$$\tilde{\mathbf{a}}^M = (\tilde{\mathbf{A}}_M^Q)^T \tilde{\mathbf{a}}^Q \in \mathbb{R}^{|A|\times 1} \tag{3.5b}$$

$$\tilde{\mathbf{A}}_Q^M = softmax({\mathbf{A}_M^Q}^T) \in \mathbb{R}^{|A|\times|Q|} \tag{3.5c}$$

Generalization module. We add a one-hop attention process before answering. We use the question representation $\tilde{\mathbf{q}}$ to query over the key memory $\overline{\mathbf{M}}^k$ via an attention mechanism, and fetch the most relevant information from the value memory, which is then used to update the question vector using a GRU [78]. Finally, we apply a residual layer [160] (i.e., $y = f(x) + x$) and batch normalization (BN) [268], which help the model performance in practice. Thus, we have

$$\hat{\mathbf{q}} = \mathrm{BN}(\tilde{\mathbf{q}} + \mathbf{q}') \tag{3.6a}$$

$$\mathbf{q}' = \mathrm{GRU}(\tilde{\mathbf{q}}, \tilde{\mathbf{m}}) \tag{3.6b}$$

$$\tilde{\mathbf{m}} = \sum_{i=1}^{|A|} a_i \cdot \tilde{\mathbf{M}}_i^v \tag{3.6c}$$

$$\mathbf{a} = \mathrm{Att}_{\mathrm{add}}^{\mathrm{GRU}}(\tilde{\mathbf{q}}, \overline{\mathbf{M}}^k) \tag{3.6d}$$

3.2.4 Answer Module

Given the representation of question Q which is $\hat{\mathbf{q}}$ and the representation of candidate answers $\{A_i\}_{i=1}^{|A|}$ which is $\{\overline{\mathbf{M}}_i^k\}_{i=1}^{|A|}$, we compute the matching score $S(\hat{\mathbf{q}}, \overline{\mathbf{M}}_i^k)$ between every pair (Q, A_i) as

$$S(\mathbf{q}, \mathbf{a}) = \mathbf{q}^T \cdot \mathbf{a} \tag{3.7}$$

which basically computes the dot product between the two input vectors. The candidate answers are then ranked by their matching scores.

3.2.5 Training and Testing

Training. Intermediate modules such as the *enhancing module* generate "premature" representations of questions (e.g., $\tilde{\mathbf{q}}$) and candidate answers (e.g., $\overline{\mathbf{M}}^k$). Even though these intermediate representations are not optimal for answer prediction, we can still use them along with the final representations to jointly train the model, which we find helps the training probably by providing more supervision since we are directly forcing intermediate representations to be helpful for prediction. Moreover, we directly match interrogative words to KB answer types. A question Q is represented by a 16-dimensional interrogative word (we use "which", "what", "who", "whose", "whom", "where", "when", "how", "why" and "whether") embedding $\mathbf{q}^w$ and a candidate answer A_i is represented by entity type embedding $\mathbf{H}_i^{t_2}$ with the same size. We then compute the matching score $S(\mathbf{q}^w, \mathbf{H}_i^{t_2})$ between them. Although we only have weak labels (e.g., incorrect answers do not necessarily imply incorrect types) for the type matching task, and there are no shared representations between two tasks, we find in practice this strategy helps the training process.

Loss function. In the training phase, we force positive candidates to have higher scores than negative candidates by using a triplet-based loss function:

$$o = g(\mathbf{H}^Q \tilde{\mathbf{a}}^Q, \sum_{j=1}^{3} \mathbf{M}_{\cdot,j}^k) + g(\tilde{\mathbf{q}}, \overline{\mathbf{M}}^k) + g(\hat{\mathbf{q}}, \overline{\mathbf{M}}^k) + g(\mathbf{q}^w, \mathbf{H}^{t_2}) \tag{3.8}$$

where $g(\mathbf{q}, \mathbf{M}) = \sum_{\substack{a^+ \in A^+ \\ a^- \in A^-}} \ell(S(\mathbf{q}, \mathbf{M}_{a^+}), S(\mathbf{q}, \mathbf{M}_{a^-}))$, and $\ell(y, \hat{y}) = \max(0, 1 + \hat{y} - y)$ is a hinge loss function, and A^+ and A^- denote the positive (i.e., correct) and negative (i.e., incorrect) answer sets, respectively. Note that at training time, the candidate answers are extracted from the KB subgraph of the gold-standard topic entity, with the memory size set to N_{max}. We adopt the following sampling strategy which works well in practice: if N_{max} is larger than the number of positive answers $|A^+|$, we keep all the positive answers and randomly select negative answers to fill up the memory; otherwise, we randomly select $\min(N_{max}/2, |A^-|)$ negative answers and fill up the remaining memory with random positive answers.

Testing. At testing time, we need to first find the topic entity. We do this by using the

top result returned by a separately trained topic entity predictor (we also compare with the result returned by the Freebase Search API). Then, the *answer module* returns the candidate answer with the highest scores as predicted answers. Since there can be multiple answers to a given question, the candidates whose scores are close to the highest score within a certain margin, θ, are regarded as good answers as well. Therefore, we formulate the inference process as follows:

$$\hat{A} = \{\hat{a} \quad | \, \hat{a} \in A \;\&\; \max_{a\prime \in A}\{S(\hat{\mathbf{q}}, \overline{\mathbf{M}}^k_{a\prime})\} - S(\hat{\mathbf{q}}, \overline{\mathbf{M}}^k_{\hat{a}}) < \theta\} \tag{3.9}$$

where $\max_{a\prime \in A}\{S(\hat{\mathbf{q}}, \overline{\mathbf{M}}^k_{a\prime})\}$ is the score of the best matched answer and $\hat{A}$ is the predicted answer set. Note that θ is a hyper-parameter which controls the degree of tolerance. Decreasing the value of θ makes the model become stricter when predicting answers.

3.2.6 Topic Entity Prediction

Given a question Q, the goal of a topic entity predictor is to find the best topic entity $\hat{c}$ from the candidate set $\{C_i\}_{i=1}^{|C|}$ returned by external topic entity linking tools (we use the Freebase Search API and S-MART [269] in our experiments). We use a convolutional network (CNN) to encode Q into a d-dimensional vector $\mathbf{e}$. For candidate topic entity C_i, we encode three types of KB aspects, namely, the entity name, entity type and surrounding relations where both entity name and type are represented as a sequence of words while surrounding relations are represented as a bag of sequences of words. Specifically, we use three CNNs to encode them into three d-dimensional vectors, namely, $\mathbf{C}_i^n$, $\mathbf{C}_i^t$ and $\mathbf{C}_i^{r_1}$. Note that for surrounding relations, we first encode each of the relations and then compute their average. Additionally, we compute an average of the relation embeddings via a relation embedding layer which we denote as $\mathbf{C}_i^{r_2}$. We then apply linear projections on the above vectors as follows:

$$\mathbf{P}_i^k = f^k([\mathbf{C}_i^n; \mathbf{C}_i^t; \mathbf{C}_i^{r_1}; \mathbf{C}_i^{r_2}]) \tag{3.10a}$$

$$\mathbf{P}_i^v = f^v([\mathbf{C}_i^n; \mathbf{C}_i^t; \mathbf{C}_i^{r_1}; \mathbf{C}_i^{r_2}]) \tag{3.10b}$$

where $\mathbf{P}_i^k$ and $\mathbf{P}_i^v$ are d-dimensional key and value representations of candidate C_i, respectively. Furthermore, we compute the updated question vector $\hat{\mathbf{e}}$ using the generalization module mentioned earlier. Next, we use a dot product to compute the similarity score between Q and C_i. A triplet-based loss function is used as formulated by $o = g(\mathbf{e}, \mathbf{P}_i^k) + g(\hat{\mathbf{e}}, \mathbf{P}_i^k)$

where $g(.)$ is the aforementioned hinge loss function. When training the predictor, along with the candidates returned from external entity linking tools, we do negative sampling (using string matching) to get more supervision. In the testing phase, the candidate with the highest score is returned as the best topic entity and no negative sampling is applied.

3.3 Experiments

This section provides an extensive evaluation of our proposed BAMnet model against state-of-the-art KBQA methods. The implementation of BAMnet is available at `https://github.com/hugochan/BAMnet`.

3.3.1 Baseline Methods

The baseline methods in our experiments include semantic parsing based methods ([3], [137] [270], [133], [128], [134], [129], [138], [130], [142], [271], [131], [132]) and information retrieval based methods ([5], [15], [9], [16], [17] and [18]).

3.3.2 Data and Metrics

We use the Freebase KB and the WebQuestions dataset, described below:

Freebase. This is a large-scale KB [12] that consists of general facts organized as subject-property-object triples. It has 41M non-numeric entities, 19K properties, and 596M assertions. In FreeBase, k-ary relations for $k > 2$ can be represented by a special entity (one for each k-tuple in the relation) and $k - 1$ binary relations (e.g., Fig. 3.2 shows an example of a 4-ary relation where the three binary relations are all connected to a dummy entity which is connected to the topic node *Ohio*). In our experiments, we do not include the dummy nodes into a candidate answer set. We also preprocess *Freebase* by removing predicates (e.g., those starting with "/common/", "/freebase", etc.) which are not related to world knowledge.

WebQuestions. This dataset [3] (`nlp.stanford.edu/software/sempre`) contains 3,778 training examples and 2,032 test examples. We further split the training instances into a training set and development set via a 80%/20% split. Each example contains three fields: the NL question, the answer list provided by Amazon Mechanical Turk (AMT) workers and the FreeBase URL page for the answers, which refers to the "gold" topic entity of the question. Approximately 85% of questions can be directly answered via a single FreeBase predicate. Also, each question can have multiple answers. In our experiments, we use a

development version of the dataset [272], which additionally provides (potentially noisy) entity mentions for each question.

Following [3], macro F1 scores (i.e., the average of F1 scores over all questions) are reported on the WebQuestions test set.

3.3.3 Model Settings

When constructing the vocabularies of words, entity types or relation types, we only consider those questions and their corresponding KB subgraphs appearing in the training and validation sets. The vocabulary size of words is $V = 100,797$. There are 1,712 entity types and 4,996 relation types in the KB subgraphs. Notably, in FreeBase, one entity might have multiple entity types. We only use the first one available, which is typically the most concrete one. For those non-entity nodes which are boolean values or numbers, we use "bool" or "num" as their types, respectively.

We also adopt a **query delexicalization** strategy where for each question, the topic entity mention as well as constraint entity mentions (i.e., those belonging to "date", "ordinal" or "number") are replaced with their types. When encoding KB context, if the overlap belongs to the above types, we also do this delexicalization, which will guarantee it matches up with the delexicalized question well in the embedding space.

Given a topic entity, we extract its 2-hop subgraph (i.e., $h = 2$) to collect candidate answers, which is sufficient for *WebQuestions*. At training time, the memory size is limited to $N_{max} = 96$ candidate answers (for the sake of efficiency). If there are more potential candidates, we do random sampling as mentioned earlier. We initialize word embeddings with pre-trained GloVe vectors [273] with word embedding size $d_v = 300$. The relation embedding size d_p, entity type embedding size d_t and hidden size d are set as 128, 16 and 128, respectively. The dropout rates on the word embedding layer, question encoder side and the answer encoder side are 0.3, 0.3 and 0.2, respectively. The batch size is set as 32, and answer module threshold $\theta = 0.7$. As for the topic entity prediction, we use the same hyperparameters. For each question, there are 15 candidates after negative sampling in the training time. When encoding a question, we use a CNN with filter sizes 2 and 3. A linear projection is applied to merge features extracted with different filters. When encoding a candidate aspect, we use a CNN with filter size 3. Linear activation and max-pooling are used together with CNNs. In the training process, we use the Adam optimizer [274] to

train the model. The initial learning rate is set as 0.001 which is reduced by a factor of 10 if no improvement is observed on the validation set in 3 consecutive epochs. The training procedure stops if no improvement is observed on the validation set in 10 consecutive epochs. The hyper-parameters are tuned on the development set.

3.3.4 Experimental Results

As shown in Table 3.1, our method can achieve an F1 score of 0.557 when the gold topic entity is known, which gives an upper bound of our model performance. When the gold topic entity is unknown, we report the results using: 1) the Freebase Search API, which achieves a recall@1 score of 0.857 on the test set for topic entity linking, and 2) the topic entity predictor, which achieves a recall@1 score of 0.898 for entity retrieval.

As for the performance of BAMnet on WebQuestions, it achieves an F1 score of 0.518 using the topic entity predictor, which is significantly better than the F1 score of 0.497 using the Freebase Search API. We can observe that BAMnet significantly outperforms previous state-of-the-art IR-based methods, which conclusively demonstrates the effectiveness of modeling bidirectional interactions between questions and the KB.

It is important to note that unlike the state-of-the-art SP-based methods, BAMnet relies on no external resources and very few hand-crafted features, but still remains competitive with those approaches. Based on careful hand-drafted rules, some SP-based methods [130], [134] can better model questions with constraints and aggregations. For example, [134] applies many manually designed rules and features to improve performance on questions with constraints and aggregations, and [130] directly models temporal (e.g., "after 2000"), ordinal (e.g., "first") and aggregation constraints (e.g., "how many") by adding detected constraint nodes to query graphs. In contrast, our method is end-to-end trainable, with very few hand-crafted rules.

Additionally, [130], [138] train their models on external Q&A datasets to get extra supervision. For a fairer comparison, we only show their results without training on external Q&A datasets. Similarly, for hyhrid systems [17], [142], we only report results without using Wikipedia free text. It is interesting to note that both [134] and [130] also use the ClueWeb dataset for learning more accurate semantics. The F1 score of [134] drops from 0.525 to 0.509 if ClueWeb information is removed. To summarize, BAMnet achieves state-of-the-art performance of 0.518 without recourse to any external resources and relies only on very few

hand-crafted features. If we assume gold-topic entities are given, then BAMnet achieves an F1 of 0.557.

Table 3.1: Results on the WebQuestions test set. Bold: best in-category performance.

Methods (ref)	Macro F_1
SP-based	
[3]	0.357
[137]	0.443
[270]	0.453
[133]	0.494
[128]	0.497
[134]	**0.525**
[129]	0.503
[138]	0.516
[130]	0.524
[142]	0.471
[271]	0.495
[131]	0.510
[132]	0.496
IR-based	
[5]	0.392
[15]	0.413
[9]	0.408
[16]	0.422
[17]	**0.471**
[18]	0.429
Our Method: BAMnet	
w/ gold topic entity	0.557
w/ Freebase Search API	0.497
w/ topic entity predictor	**0.518**

3.3.5 Ablation Study

We now discuss the performance impact of the different modules and strategies in BAMnet. Note that gold topic entity is assumed to be known when we do this ablation study, because the error introduced by topic entity prediction might reduce the real performance impact of a module or strategy. As shown in Table 3.2, significant performance drops were observed after turning off some key attention modules, which confirms that the real power of our method comes from the idea of hierarchical two-way attention. As we can

see, when turning off the *two-layered bidirectional attention network*, the model performance drops from 0.557 to 0.534. Among all submodules in the attention network, the *importance module* is the most significant since the F1 score drops to 0.540 without it, thereby confirming the effectiveness of modeling the query-to-KB attention flow. On the flip side, the importance of modeling the KB-to-query attention flow is confirmed by the fact that replacing the *KB-aware attention module* with self-attention significantly degrades the performance. Besides, the secondary attention layer, the *enhancing module*, also contributes to the overall model performance. Finally, we find that the topic entity delexicalization strategy has a big influence on the model performance while the constraint delexicalization strategy only marginally boosts the performance.

Table 3.2: Ablation results on the WebQuestions test set. Gold topic entity is assumed to be known.

Methods	Macro F_1
all	**0.557**
w/o two-layered bidirectional attn	0.534
w/o kb-aware attn (+self-attn)	0.544
w/o importance module	0.540
w/o enhancing module	0.550
w/o generalization module	0.542
w/o joint type matching	0.545
w/o topic entity delexicalization	0.529
w/o constraint delexicalization	0.554

3.3.6 Interpretability Analysis

Here, we show that our method does capture the mutual interactions between question words and KB aspects, by visualizing the attention matrix $\mathbf{A}^{QM}$ produced by the *reasoning module*. Fig. 3.4 shows the attention heatmap generated for a test question "who did location surrender to in __number__" (where "location" and "__number__" are entity types which replace the topic entity mention "France" and the constraint entity mention "ww2", respectively in the original question). As we can see, the attention network successfully detects the interactions between "who" and answer type, "surrender to" and answer path, and focuses more on those words when encoding the question.

To further examine the importance of the two-way flow of interactions, in Table 3.3,

we show the predicted answers of BAMnet with and without the *two-layered bidirectional attention network* on samples questions from the WebQuestions test set. We divide the questions into three categories based on which kind of KB aspect is the most crucial for answering them. As we can see, compared to the simplified version which is not equipped with bidirectional attention, our model is more capable of answering all the three types of questions.

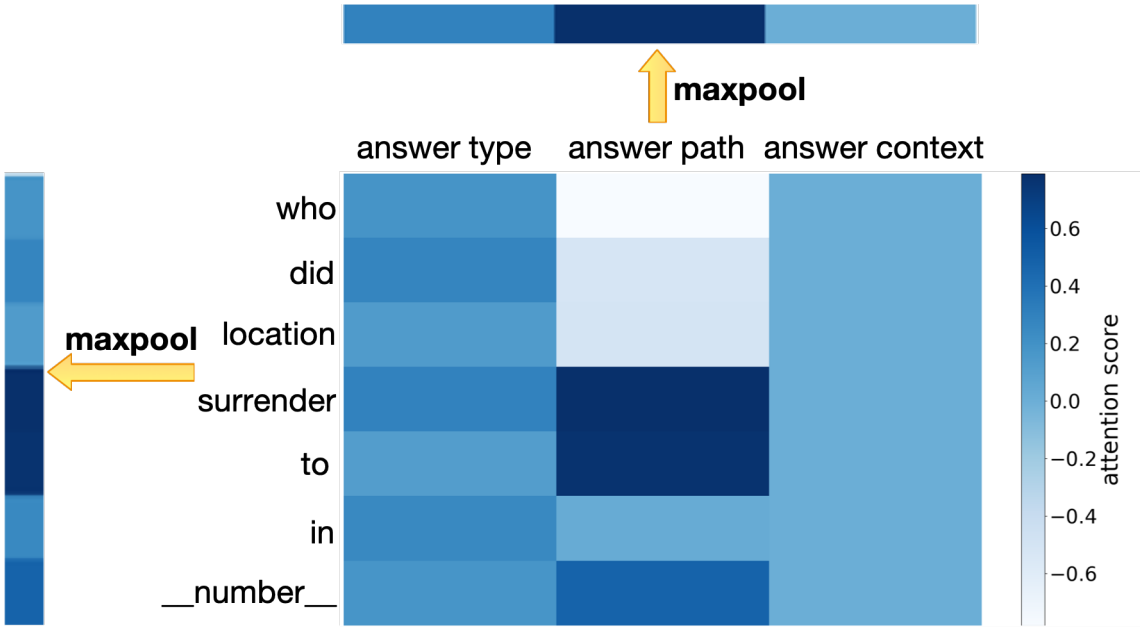

Fig. 3.4: Attention heatmap generated by the reasoning module. Best viewed in color.

3.3.7 Error Analysis

To better examine the limitations of our approach, we randomly sampled 100 questions on which our method performed poorly (i.e., with per-question F1 score less than 0.6), and categorized the errors. We found that around 33% of errors are due to label issues of gold answers and are not real mistakes. This includes incomplete and erroneous labels, and also alternative correct answers. For instance our method generated more complete answers (i.e., "Modern art", "Pop art", "Printmaking", "Photography" and "Painting") for the question "what types of art did andy warhol do?" while the gold standard answer is just "Pop art". Likewise, the gold standard answer for "where are samsung based?" is "Seoul" while our method answered "Seoul", "Seoul Capital Area" and "Daegu" that are all correct. Constraints are another source of errors (11%), with temporal constraints accounting for

Table 3.3: Predicted answers of BAMnet w/ and w/o bidirectional attention on the WebQuestions test set.

KB Aspects	Questions	BAMnet w/o BiAttn.	BAMnet	Gold Answers
Answer Type	What degrees did Obama get in college?	Harvard Law School, Columbia University, Occidental College	Bachelor of Arts, Juris Doctor, Political Science	Juris Doctor, Bachelor of Arts
	What music period did Beethoven live in?	Austrian Empire, Germany, Bonn	Classical music, Opera	Opera, Classical music
Answer Path	Where did Queensland get its name from?	Australia	Queen Victoria	Queen Victoria
	Where does Delaware river start?	Delaware Bay	West Branch Delaware River, Mount Jefferson	West Branch Delaware River, Mount Jefferson
Answer Context	What are the major cities in Ukraine?	Kiev, Olyka, ... Vynohradiv, Husiatyn	Kiev	Kiev
	Who is running for vice president with Barack Obama 2012?	David Petraeus	Joe Biden	Joe Biden

most. Some questions have implicit temporal (e.g., tense) constraints which our method does not model. For instance, "where do the seattle seahawks play?" is being asked in present tense, but our method generates more answers, which would be correct if tense is ignored. An example of an ordinal question, which our method failed to answer is "what was pink floyd's first album?" A third source of error is what we term type errors (13%), for which our method generates more answers than needed because of poorly utilizing answer type information. For instance, for "where did derek fisher go to college?", some answers generated by our method are actually high schools. This can be addressed to some extent by tuning the answer module threshold θ, which can improve the precision. For some questions, our method just failed to figure out the correct answer type. Lexical gap is another source of errors (5%). For instance, the keyword "electorate" in question "what electorate does anna bligh represent?" neither appears in the training set nor the corresponding KB subgraph of topic entity "anna bligh". Finally, other sources of errors (38%) include topic entity prediction error, question ambiguity, incomplete answers and other miscellaneous errors.

3.4 Conclusion and Future Work

We introduced a novel and effective bidirectional attentive memory network for the purpose of KBQA. To our best knowledge, we are the first to model the mutual interactions

between questions and a KB, which allows us to distill the information that is the most relevant to answering the questions on both sides of the question and KB. Experimental results show that our method significantly outperforms previous IR-based methods while remaining competitive with hand-crafted SP-based methods. Both ablation study and interpretability analysis verify the effectiveness of the idea of modeling mutual interactions. In addition, our error analysis shows that our method actually performs better than what the evaluation metrics indicate.

In the future, we would like to explore effective ways of modeling more complex types of constraints (e.g., ordinal, comparison and aggregation).

CHAPTER 4
CONVERSATIONAL MACHINE READING COMPREHENSION

4.1 Overview

Conversational machine comprehension (MC) has proven significantly more challenging compared to traditional MC since it requires better utilization of conversation history [37], [38], [40]. However, most existing approaches [41]–[43] do not effectively capture conversation history and thus have trouble handling questions involving coreference or ellipsis. Moreover, when reasoning over passage text, most of them simply treat it as a word sequence without exploring rich semantic relationships among words [150], [151]. In this work, we first propose a simple yet effective graph structure learning technique to dynamically construct a question and conversation history aware context graph at each conversation turn. Then we propose a novel Recurrent Graph Neural Network (RGNN), and based on that, we introduce a flow mechanism to model the temporal dependencies in a sequence of context graphs. Answers are finally predicted based on the matching score of the question embedding and the context graph embedding at each turn.

We highlight our contributions as follows:

- We propose a novel GNN based model, namely GraphFlow, for conversational MC which captures conversational flow in a dialog.
- We dynamically construct a question and conversation history aware context graph at each turn, and propose a novel Recurrent Graph Neural Network based flow mechanism to process a sequence of context graphs.
- On three public benchmarks (i.e., CoQA, QuAC and DoQA), our model shows competitive performance compared to existing state-of-the-art methods. In addition, visualization experiments show that our model can offer good interpretability for the reasoning process.

This chapter previously appeared as: Y. Chen, L. Wu, and M. J. Zaki, "Graphflow: Exploiting conversation flow with graph neural networks for conversational machine comprehension," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Jul. 2020, pp. 1230–1236.

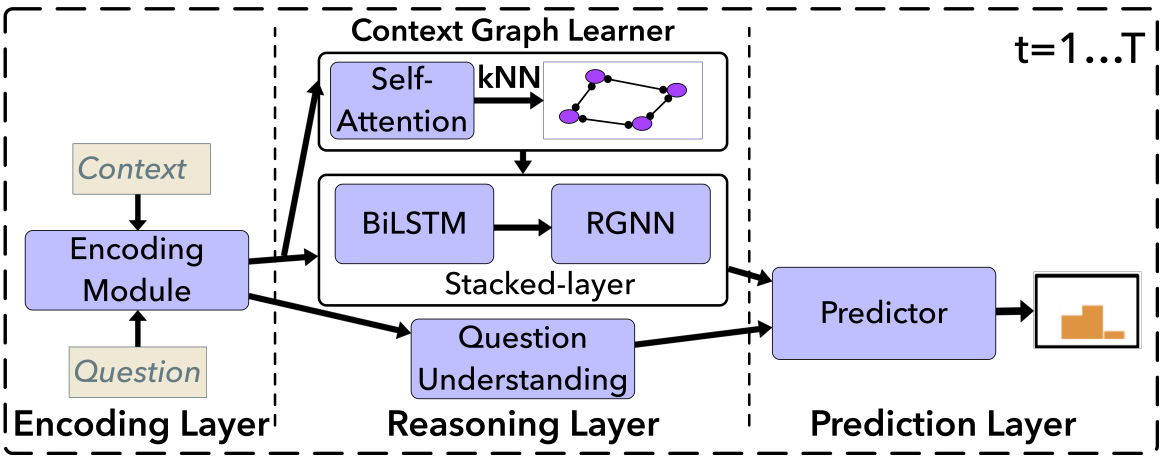

Fig. 4.1: Overall architecture of the proposed model.

4.2 Approach: GraphFlow

The task of conversational Machine Comprehension is to answer a natural language question given the context and conversation history. Let us denote C as the context which consists of a word sequence $\{c_1, c_2, ..., c_m\}$ and $Q^{(i)}$ as the question at the i-th turn which consists of a word sequence $\{q_1^{(i)}, q_2^{(i)}, ..., q_n^{(i)}\}$. And there are totally T turns in a conversation.

As shown in Fig. 4.1, our proposed GraphFlow model consists of *Encoding Layer*, *Reasoning Layer* and *Prediction Layer*. The *Encoding Layer* encodes conversation history and context that aligns question information. The *Reasoning Layer* dynamically constructs a question and conversation history aware context graph at each turn, and then applies a flow mechanism to process a sequence of context graphs. The *Prediction Layer* predicts the answers based on the matching score of the question embedding and the context graph embedding. The details of these modules are given next.

4.2.1 Encoding Layer

We apply an effective encoding layer to encode the context and the question, which additionally exploits conversation history and interactions between them.

Linguistic features. For context word c_j, we encode linguistic features into a vector $f_{\text{ling}}(c_j^{(i)})$ concatenating POS (part-of-speech), NER (named entity recognition) and exact matching (which indicates whether c_j appears in $Q^{(i)}$) embeddings.

Pretrained word embeddings. We use 300-dim GloVe [273] embeddings and 1024-dim BERT [123] embeddings to embed each word in the context and the question. Compared to

GloVe, BERT better utilizes contextual information when embedding words.

Aligned question embeddings. Exact matching matches words on the surface form; we further apply an attention mechanism to learn soft alignment between context words and question words. Since this soft alignment operation is conducted in parallel at each turn, for the sake of simplicity, we omit the turn index i when formulating the alignment operation. Following [275], for context word c_j at each turn, we incorporate an aligned question embedding

$$f_{align}(c_j) = \sum_k \beta_{j,k} \mathbf{g}_k^Q \tag{4.1}$$

where $\mathbf{g}_k^Q$ is the GloVe embedding of the k-th question word q_k and $\beta_{j,k}$ is an attention score between context word c_j and question word q_k. The attention score $\beta_{j,k}$ is computed by

$$\beta_{j,k} \propto \exp(\mathrm{ReLU}(\mathbf{W}\mathbf{g}_j^C)^T \mathrm{ReLU}(\mathbf{W}\mathbf{g}_k^Q)) \tag{4.2}$$

where $\mathbf{W}$ is a $d \times 300$ trainable weight with d being the hidden state size, and $\mathbf{g}_j^C$ is the GloVe embedding of context word c_j. To simplify notation, we denote the above attention mechanism as $\mathrm{Align}(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$, meaning that an attention matrix is computed between two sets of vectors $\mathbf{X}$ and $\mathbf{Y}$, which is later used to get a linear combination of vector set $\mathbf{Z}$. Hence we can reformulate the above alignment as

$$f_{\mathrm{align}}(C) = \mathrm{Align}(\mathbf{g}^C, \mathbf{g}^Q, \mathbf{g}^Q) \tag{4.3}$$

Conversation history. Conversation history is crucial for understanding questions. Following [38], we concatenate a feature vector $f_{\mathrm{ans}}(c_j^{(i)})$ encoding previous N answer locations to context word embeddings. Preliminary experiments showed that it is helpful to also prepend previous N question-answer pairs to a current question. When prepending conversation history, a common choice is to separate the current question from conversation history using certain special token, which does not work well in practice as observed by [38]. We find a more effective strategy to do the separation which is for each word vector in an augmented question, we concatenate a turn marker embedding $f_{\mathrm{turn}}(q_k^{(i)})$ indicating which turn the word belongs to (e.g., i indicates the previous i-th turn).

In summary, at the i-th turn in a conversation, each context word c_j is encoded by a vector $\mathbf{w}_{c_j}^{(i)}$ which is a concatenation of linguistic vector $f_{\mathrm{ling}}(c_j^{(i)})$, word embeddings (i.e., $\mathbf{g}_j^C$

and BERT_j^C), aligned vector $f_{\text{align}}(c_j^{(i)})$ and answer vector $f_{\text{ans}}(c_j^{(i)})$. And each question word $q_k^{(i)}$ is encoded by a vector $\mathbf{w}_{q_k}^{(i)}$ which is a concatenation of word embeddings (i.e., $\mathbf{g}_k^{Q^{(i)}}$ and $\text{BERT}_k^{Q^{(i)}}$) and turn marker vector $f_{\text{turn}}(q_k^{(i)})$. We denote $\mathbf{W}_C^{(i)}$ and $\mathbf{W}_Q^{(i)}$ as a sequence of context word vectors $\mathbf{w}_{c_j}^{(i)}$ and question word vectors $\mathbf{w}_{q_k}^{(i)}$, respectively.

4.2.2 Reasoning Layer

When performing reasoning over context, unlike most previous methods that regard context as a word sequence, we opt to treat context as a "graph" of words that captures rich semantic relationships among words, and apply a Recurrent Graph Neural Network to process a sequence of context graphs.

4.2.2.1 *Question Understanding*

For a question $Q^{(i)}$, we apply a bidirectional LSTM [172] to the question embeddings $\mathbf{W}_Q^{(i)}$ to obtain contextualized embeddings $\mathbf{Q}^{(i)} \in \mathbb{R}^{d \times n}$.

$$\mathbf{Q}^{(i)} = \mathbf{q}_1^{(i)}, ..., \mathbf{q}_n^{(i)} = \text{BiLSTM}(\mathbf{W}_Q^{(i)}) \tag{4.4}$$

And the question is then represented as a weighted sum of question word vectors via a self attention mechanism,

$$\widetilde{\mathbf{q}}^{(i)} = \sum_k a_k^{(i)} \mathbf{q}_k^{(i)} \tag{4.5a}$$

$$\text{where} \quad a_k^{(i)} \propto \exp(\mathbf{w}^T \mathbf{q}_k^{(i)}) \tag{4.5b}$$

where $\mathbf{w}$ is a d-dim trainable weight.

Finally, to capture the dependency among question history, we encode the sequence of questions with a LSTM to generate history-aware question vectors.

$$\mathbf{p}^{(1)}, ..., \mathbf{p}^{(T)} = \text{LSTM}(\widetilde{\mathbf{q}}^{(1)}, ..., \widetilde{\mathbf{q}}^{(T)}) \tag{4.6}$$

The output hidden states of the LSTM network $\mathbf{p}^{(1)}, ..., \mathbf{p}^{(T)}$ will be used for predicting answers.

4.2.2.2 *Context Graph Learning*

The intrinsic context graph structure is unfortunately unknown. Moreover, the context graph structure might vary across different turns by considering the changes of questions and conversation history. Most existing applications of GNNs [225], [226], [232] use ground-truth or manually constructed graphs which have some limitations. First, the ground-truth graphs are not always available. Second, errors in manual construction process can be propagated to subsequent modules. Unlike previous methods, we automatically construct graphs from raw context, which are combined with the rest of the system to make the whole learning system end-to-end trainable. We dynamically build a question and conversation history aware context graph to model semantic relationships among context words at each turn.

Specifically, we first apply an attention mechanism to the context representations $\mathbf{W}_C^{(i)}$ (which additionally incorporate both question information and conversation history) at the i-th turn to compute an attention matrix $\mathbf{A}^{(i)}$, serving as a weighted adjacency matrix for the context graph, defined as,

$$\mathbf{A}^{(i)} = (\mathbf{W}_C^{(i)} \odot \mathbf{u})^T \mathbf{W}_C^{(i)} \tag{4.7}$$

where $\odot$ denotes element-wise multiplication, and $\mathbf{u}$ is a non-negative d_c-dim trainable weight vector which learns to highlight different dimensions of $\mathbf{w}_{c_j}^{(i)}$ whose dimension is d_c.

Considering that a fully connected context graph is not only computationally expensive but also might introduce noise (i.e., unimportant edges), a simple kNN-style graph sparsification operation is applied to select the most important edges from the fully connected graph, resulting in a sparse graph. To be concrete, given a learned attention matrix $\mathbf{A}^{(i)}$, we only keep the K nearest neighbors (including itself) as well as the associated attention scores (i.e., the remaining attentions scores are masked off) for each context node. We then apply a softmax function to these selected adjacency matrix elements to get a normalized adjacency matrix.

$$\widetilde{\mathbf{A}}^{(i)} = \text{softmax}(\text{topk}(\mathbf{A}^{(i)})) \tag{4.8}$$

Note that the supervision signal is still able to back-propagate through the kNN-style graph sparsification module since the K nearest attention scores are kept and used to compute the weights of the final normalized adjacency matrix.

To summarize, at each turn in a conversation, we dynamically build a weighted directed

context graph $\mathcal{G}^{(i)}$ which depends on the semantic meanings of the context, the question as well as the conversation history.

4.2.2.3 *Context Graph Reasoning*

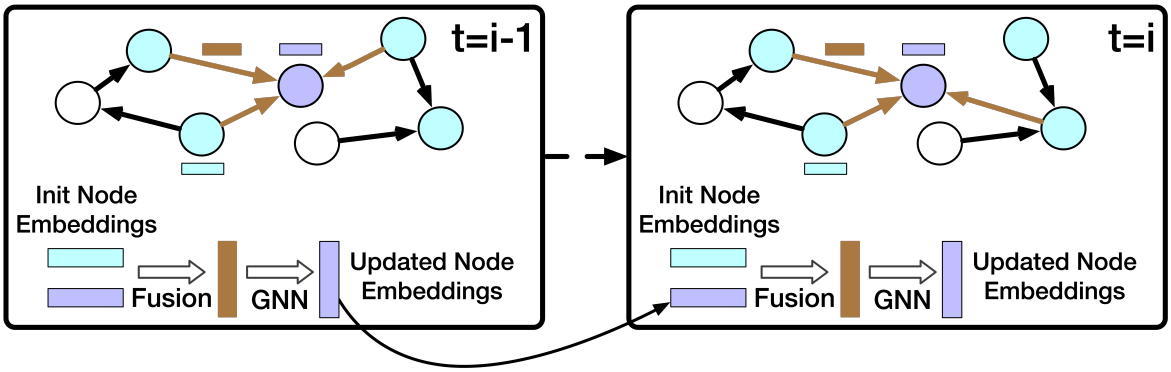

Fig. 4.2: Architecture of the proposed Recurrent Graph Neural Network for processing a sequence of context graphs.

When reasoning over a sequence of context graphs, we want to consider not only the relationships among graph nodes, but also the sequential dependencies among graphs. Especially for the conversational MC task, we hope the results of previous reasoning processes can be incorporated into the current reasoning process since they potentially capture important information for answering the current question.

Therefore, we propose a novel *Recurrent Graph Neural Network* (RGNN) to process a sequence of graphs, as shown in Fig. 4.2. As we advance in a sequence of graphs, we process each graph using a shared GNN cell and the GNN output will be used when processing the next graph. One can think that it is analogous to an RNN-style structure where the main difference is that each element in a sequence is not a data point, but instead a graph. Our RGNN module combines the advantages of RNNs which are good at sequential learning (i.e., modeling sequential data), and GNNs which are good at relational reasoning (i.e., modeling graph-structured data).

The computational details of RGNN are as follows. Let us denote $\mathbf{C}^{(i)}$ as the initial context node embedding at the i-th turn. Before we apply a GNN to the context graph $\mathcal{G}^{(i)}$, we update its node embeddings by fusing both the original node information $\mathbf{C}^{(i)}$ and the updated node information $\overline{\mathbf{C}}^{(i-1)}$ computed by a parameter-sharing GNN at the $(i-1)$-th

turn via a fusion function,

$$\overline{\mathbf{C}}^{(i)} = \text{GNN}(\text{Fuse}(\mathbf{C}^{(i)}, \overline{\mathbf{C}}^{(i-1)}), \widetilde{\mathbf{A}}^{(i)}) \tag{4.9}$$

where we set $\overline{\mathbf{C}}^{(0)} = \mathbf{C}^0$ as we do not incorporate any historical information at the first turn. The fusion function is designed as a gated sum of two information sources,

$$\text{Fuse}(\mathbf{a}, \mathbf{b}) = \mathbf{z} * \mathbf{a} + (1 - \mathbf{z}) * \mathbf{b} \tag{4.10a}$$

$$\mathbf{z} = \sigma(\mathbf{W}_z[\mathbf{a}; \mathbf{b}; \mathbf{a} * \mathbf{b}; \mathbf{a} - \mathbf{b}] + \mathbf{b}_z) \tag{4.10b}$$

where σ is a sigmoid function and $\mathbf{z}$ is a gating vector. As a result, the graph node embedding outputs of the reasoning process at the previous turn are used as a starting state when reasoning at the current turn.

We use Gated Graph Neural Networks (GGNN) [197] as our GNN cell, but the framework is agnostic to the particular choice of GNN cell. In GGNN we do multi-hop message passing through a graph to capture long-range dependency where the same set of network parameters are shared at every hop of computation. At each hop of computation, for every graph node, we compute an aggregation vector as a weighted average of all its neighboring node embeddings where the weights come from the normalized adjacency matrices $\widetilde{\mathbf{A}}^{(i)}$. Then, a Gated Recurrent Unit (GRU) [78] is used to update node embeddings by incorporating the aggregation vectors. We use the updated node embeddings at the last hop as the final node embeddings.

To simplify notation, we denote the above RGNN module as $\overline{\mathbf{C}}^{(i)} = \text{RGNN}(\mathbf{C}^{(i)}, \widetilde{\mathbf{A}}^{(i)})$, $i = 1, \ldots, T$ which takes as input a sequence of graph node embeddings $\{\mathbf{C}^{(i)}\}_{i=1}^{T}$ as well as a sequence of the normalized adjacency matrices $\{\widetilde{\mathbf{A}}^{(i)}\}_{i=1}^{T}$, and outputs a sequence of updated graph node embeddings $\{\overline{\mathbf{C}}^{(i)}\}_{i=1}^{T}$.

While a GNN is responsible for modeling global interactions among context words, modeling local interactions between consecutive context words is also important for the task. Therefore, before feeding the context word representations to a GNN, we first apply a BiLSTM to encode local dependency, that is, $\mathbf{C}^{(i)} = \text{BiLSTM}(\mathbf{W}_C^{(i)})$, and then use the output $\mathbf{C}^{(i)}$ as the initial context node embedding.

Inspired by recent work [276] on modeling the context with different levels of granu-

larity, we choose to apply stacked RGNN layers where one RGNN layer is applied on low level representations of the context and the second RGNN layer is applied on high level representations of the context. The output of the second RGNN layer $\{\widetilde{\mathbf{C}}^{(i)}\}_{i=1}^{T}$ is the final context representations.

$$\mathbf{H}_C^{(i)} = [\overline{\mathbf{C}}^{(i)}; \mathbf{g}^C; \text{BERT}^C] \tag{4.11a}$$

$$\mathbf{H}_Q^{(i)} = [\mathbf{Q}^{(i)}; \mathbf{g}^{Q^{(i)}}; \text{BERT}^{Q^{(i)}}] \tag{4.11b}$$

$$f^2_{\text{align}}(C^{(i)}) = \text{Align}(\mathbf{H}_C^{(i)}, \mathbf{H}_Q^{(i)}, \mathbf{Q}^{(i)}] \tag{4.11c}$$

$$\hat{\mathbf{C}}^{(i)} = \text{BiLSTM}([\overline{\mathbf{C}}^{(i)}; f^2_{\text{align}}(C^{(i)})]) \tag{4.11d}$$

$$\widetilde{\mathbf{C}}^{(i)} = \text{RGNN}(\hat{\mathbf{C}}^{(i)}, \widetilde{\mathbf{A}}^{(i)}),\ i = 1, \ldots, T \tag{4.11e}$$

4.2.3 Prediction Layer

We predict answer spans by computing the start and end probabilities of the j-th context word for the i-th question. For the sake of simplicity, we omit the turn index i when formulating the prediction layer. The start probability P_j^S is calculated by,

$$P_j^S \ \propto\ \exp(\widetilde{\mathbf{c}}_j^T \mathbf{W}_S \mathbf{p}) \tag{4.12}$$

where $\mathbf{W}_S$ is a $d \times d$ trainable weight and $\mathbf{p}$ (turn index omitted) is the question representation obtained in Eq. (4.6). Next, $\mathbf{p}$ is passed to a GRU cell by incorporating context summary and converted to $\widetilde{\mathbf{p}}$.

$$\widetilde{\mathbf{p}} = \text{GRU}(\mathbf{p}, \sum_j P_j^S \widetilde{\mathbf{c}}_j) \tag{4.13}$$

Then, the end probability P_j^E is calculated by,

$$P_j^E \ \propto\ \exp(\widetilde{\mathbf{c}}_j^T \mathbf{W}_E \widetilde{\mathbf{p}}) \tag{4.14}$$

where $\mathbf{W}_E$ is a $d \times d$ trainable weight.

We apply an answer type classifier to handle unanswerable questions and questions whose answers are not text spans in the context. The probability of the answer type (e.g., "unknown", "yes" and "no") is calculated as follows,

$$P^C = \sigma(f_c(\mathbf{p})[f_{\text{mean}}(\widetilde{\mathbf{C}}); f_{\max}(\widetilde{\mathbf{C}})]^T) \tag{4.15}$$

where f_c is a dense layer which maps a d-dim vector to a (num_class $\times$ $2d$)-dim vector. Further, σ is a sigmoid function for binary classification and a softmax function for multi-class classification. $f_{\text{mean}}(.)$ and $f_{\max}(.)$ denote the average pooling and max pooling operations, respectively.

4.2.4 Training and Testing

The training objective for the i-th turn is defined as the cross entropy loss of both text span prediction (if the question requires it) and answer type prediction where the turn index i is omitted for the sake of simplicity,

$$\mathcal{L} = -I^S(log(P_s^S) + log(P_e^E)) + \log P_t^C \tag{4.16}$$

where I^S indicates whether the question requires answer span prediction, s and e are the ground-truth start and end positions of the span, and t indicates the ground-truth answer type.

During inference, we first use P^C to predict whether the question requires text span prediction. If yes, we predict the span to be $\hat{s}, \hat{e}$ with maximum $P_{\hat{s}}^S, P_{\hat{e}}^E$ subject to certain maximum span length threshold.

4.3 Experiments

In this section, we conduct an extensive evaluation of our proposed model against state-of-the-art conversational MC models on various benchmarks. The implementation of our model is publicly available at `https://github.com/hugochan/GraphFlow`.

4.3.1 Baseline Methods

We compare our method with the following baselines: i) PGNet [41], ii) DrQA [42], iii) DrQA+PGNet [37], iv) BiDAF++ [43], v) FlowQA [151], vi) SDNet [150], vii) BERT [123] and viii) Flow (unpublished). Detailed descriptions of the baselines are provided next.

PGNet is a pointer-generator network equipped with the coverage mechanism which was originally designed for the abstractive text summarization task.

DrQA is strong machine comprehension baseline that combines a search component based on bigram hashing and TF-IDF matching with a multi-layer RNN model for detecting answers

from passages.

DrQA+PGNet is a hybrid model in which DrQA first points to the answer evidence in the passage, and PGNet naturalizes the evidence into an answer.

BiDAF++ is a model based on BiDAF [8], augmented with self attention [277] and ELMo contextualized embeddings [122].

FlowQA is a flow-based model that can incorporate intermediate representations generated during the process of answering previous questions, through an alternating parallel processing structure.

SDnet is a contextual attention-based deep neural network which leverages inter-attention and self-attention on passage and conversation history in order to capture the dialog flow.

BERT is a large-scale pretrained language model based on bidirectional Transformer encoders. It has been shown that finetuning the BERT model on downstream tasks can achieve the state-of-the-art results on various NLP tasks [123].

Flow is an unpublished model for conversational machine comprehension.

Following previous works [150], [151], we use an extractive approach with answer type classifiers on all benchmarks. To handle different answer types in CoQA, we predict the probability distribution of the answer type (SPAN, YES, NO, and UNANSWERABLE) and replace the predicted span with "yes", "no", or "unknown" tokens except for the "SPAN" answer type. In QuAC, the unanswerable questions are handled as an answer span (P contains a special token), and the type prediction for yes/no questions is not evaluated on the leaderboard. Therefore, we skip the answer type prediction step.

4.3.2 Data and Metrics

CoQA [37] contains 127k questions with answers, obtained from 8k conversations. Answers are in free-form and hence are not necessarily text spans from context (i.e., 33.2% of the questions have abstractive answers). Although this calls for a generation approach, following previous work [43], [150], [151], as detailed in Section 4.2.3, we adopt an extractive approach with additional answer type classifiers to handle non-extractive questions. The average length of questions is only 5.5 words, which means conversation history is important for better understanding those questions. The average number of turns per dialog is 15.2. Notably, in the testing set, there are two out-of-domain datasets which are reserved for testing

only. QuAC [38] contains 98k questions with answers, obtained from 13k conversations. All the answers are text spans from context. The average length of questions is 6.5 and there are on average 7.2 questions per dialog. The average length of QuAC context is 401 which is longer than that of CoQA which is 271. The average length of QuAC answers is 14.6 which is also longer than that of CoQA which is 2.7. DoQA [40] contains 7.3k questions with answers, obtained from 1.6k conversations in the cooking domain. Similar to CoQA, 31.3% of the answers are not directly extracted from context.

The main evaluation metric is F1 score which is the harmonic mean of precision and recall at word level between the predication and ground truth. In addition, for QuAC and DoQA, the Human Equivalence Score (HEQ) is used to judge whether a system performs as well as an average human. HEQ-Q and HEQ-D are model accuracies at question level and dialog level. Please refer to [37], [38] for details of these metrics.

4.3.3 Model Settings

We construct the vocabulary of words from the training set but filter out those infrequent words (i.e., word count less than 5) to reduce the vocabulary size. The embedding sizes of POS, NER, exact matching and turn marker embeddings are set to 12, 8, 3 and 3, respectively. We fix the pretrained GloVe vectors. Following [150], we pre-compute BERT embeddings for each word using a weighted sum of BERT layer outputs. The size of all hidden layers is set to 300. When constructing context graphs, the neighborhood size is set to 10. The number of GNN hops is set to 5 for CoQA and DoQA, and 3 for QuAC. During training, we apply dropout after embedding layers (0.3 for GloVe and 0.4 for BERT) and RNN layers (0.3 for all). We use Adamax [274] as the optimizer and the learning rate is set to 0.001. We reduce the learning rate by a factor of 0.5 if the validation F1 score has stopped improving every one epoch. We stop the training when no improvement is seen for 10 consecutive epochs. We clip the gradient at length 10. We batch over dialogs and the batch size is set to 1. When augmenting the current turn with conversation history, we only consider the previous two turns. When doing text span prediction, the span is constrained to have a maximum length of 12 for CoQA, 35 for QuAC and 30 for DoQA. All these hyper-parameters are tuned on the development set.

4.3.4 Experimental Results

As shown in Table 4.1, Table 4.2, and Table 4.3, our model outperforms or achieves competitive performance compared with various state-of-the-art baselines. Compared with FlowQA which is also based on the flow idea, our model improves F1 by 2.3% on CoQA, 0.8% on QuAC and 2.5% on DoQA, which demonstrates the superiority of our RGNN based flow mechanism over the IF mechanism. Compared with SDNet which relies on sophisticated inter-attention and self-attention mechanisms, our model improves F1 by 0.7% on CoQA.

Table 4.1: Model and human performance (% in F1 score) on the CoQA test set.

	Child.	Liter.	Mid-High.	News	Wiki	Reddit	Science	Overall
PGNet	49.0	43.3	47.5	47.5	45.1	38.6	38.1	44.1
DrQA	46.7	53.9	54.1	57.8	59.4	45.0	51.0	52.6
DrQA+PGNet	64.2	63.7	67.1	68.3	71.4	57.8	63.1	65.1
BiDAF++	66.5	65.7	70.2	71.6	72.6	60.8	67.1	67.8
FlowQA	73.7	71.6	76.8	79.0	80.2	67.8	76.1	75.0
Flow	–	–	–	–	–	–	–	75.8
SDNet	75.4	73.9	77.1	**80.3**	**83.1**	69.8	76.8	76.6
GraphFlow	**77.1**	**75.6**	**77.5**	79.1	82.5	**70.8**	**78.4**	**77.3**
Human	90.2	88.4	89.8	88.6	89.9	86.7	88.1	88.8

Table 4.2: Model and human performance (in %) on the QuAC test set.

	F1	HEQ-Q	HEQ-D
BiDAF++	60.1	54.8	4.0
FlowQA	64.1	59.6	**5.8**
GraphFlow	**64.9**	**60.3**	5.1
Human	80.8	100	100

Table 4.3: Model and human performance (in %) on the DoQA test set.

	F1	HEQ-Q	HEQ-D
BERT	41.4	38.6	4.8
FlowQA	42.8	35.5	5.0
GraphFlow	**45.3**	**41.5**	**5.3**
Human	86.7	–	–

Table 4.4: Ablation study (in %) on the CoQA dev. set.

	F1
GraphFlow (2-His)	**78.3**
– PreQues	78.2
– PreAns	77.7
– PreAnsLoc	76.6
– BERT	76.0
– RecurrentConn	69.9
– RGNN	68.8
– kNN	69.9
GraphFlow (1-His)	78.2
GraphFlow (0-His)	76.7

4.3.5 Ablation Study

We conduct an extensive ablation study to further investigate the performance impact of different components in our model. Here we briefly describe ablated systems: – RecurrentConn removes temporal connections between consecutive context graphs, – RGNN removes the RGNN module, – kNN removes the kNN-style graph sparsification operation, – PreQues does not prepend previous questions to the current turn, – PreAns does not prepend previous answers to the current turn, – PreAnsLoc does not mark previous answer locations in the context, and – BERT removes pretrained BERT embeddings. We also show the model performance with no conversation history GraphFlow (0-His) or one previous turn of the conversation history GraphFlow (1-His).

Table 4.4 shows the contributions of the above components on the CoQA development set. Our proposed RGNN module contributes significantly to the model performance (i.e., improves F1 score by 7.2%). In addition, within the RGNN module, both the GNN part (i.e., 1.1% F1) and the temporal connection part (i.e., 6.1% F1) contribute to the results. This verifies the effectiveness of representing a passage as a graph and modeling the temporal dependencies in a sequence of context graphs. The kNN-style graph sparsification operation also contributes significantly to the model performance. We notice that explicitly adding conversation history to the current turn helps the model performance. We can see that the previous answer information is more crucial than the previous question information. And among many ways to use the previous answer information, directly marking previous answer locations seems to be the most effective. We conjecture this is partially because the turn transitions in a conversation are usually smooth and marking the previous answer

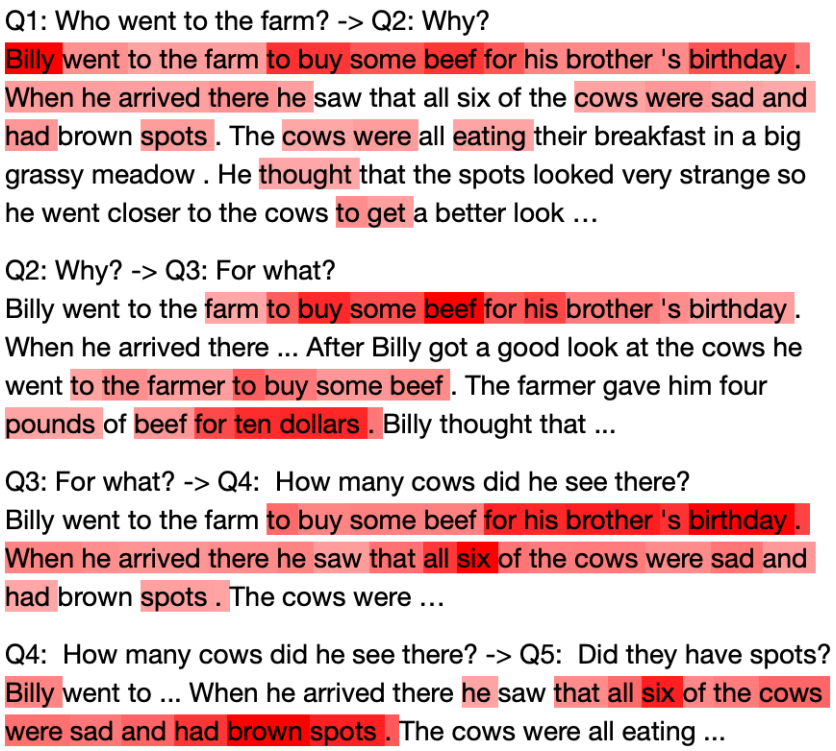

Fig. 4.3: The highlighted part of the context indicates GraphFlow's focus shifts between consecutive question turns.

locations helps the model better identify relevant context chunks for the current question. Last but not least, we find that the pretrained BERT embedding has significant impact on the performance, which demonstrates the power of large-scale pretrained language models.

4.3.6 Interpretability Analysis

Following [151], we visualize the changes of hidden representations of context words between consecutive turns. Specifically, we compute cosine similarity of hidden representations of the same context words at consecutive turns, and then highlight the words that have small cosine similarity scores (i.e., change more significantly). For better visualization, we apply an attention threshold of 0.3 to highlight only the dramatically changing context words. Fig. 4.3 highlights the most changing context words (due to the page limit, we do not show full context) between consecutive turns in a conversation from the CoQA dev. set. As we can see, the hidden representations of context words which are relevant to the consecutive questions are changing most and thus highlighted most. We suspect this is in part because when the focus shifts, the model finds out that the context chunks relevant to the previous

turn become less important but those relevant to the current turn become more important. Therefore, the memory updates in these regions are the most active.

4.4 Conclusion and Future Work

We proposed a novel Graph Neural Network (GNN) based model, namely GraphFlow, for conversational machine comprehension (MC) which carries over the reasoning output throughout a conversation. Besides, we proposed a simple yet effective graph structure learning technique to dynamically construct a question and conversation history aware context graph at each conversation turn. On three recently released conversational MC benchmarks, our proposed model achieves competitive results compared with previous approaches. Interpretability analysis shows that our model can offer good interpretability for the reasoning process.

In the future, we would like to investigate more effective ways of automatically learning graph structures from free text and modeling temporal connections between sequential graphs.

CHAPTER 5
NATURAL QUESTION GENERATION FROM KGS

5.1 Overview

The task of question generation (QG) aims to generate natural language questions based on a given form of data, such as KG or tables [59], [63], text [73], [75], [152], or images [57], where the generated questions need to be answerable from the input data. In this paper, we focus on QG from a KG subgraph.

Knowledge graph (KG) question generation (QG) aims to generate natural language questions from KGs and target answers. Previous works mostly focus on a simple setting which is to generate questions from a single KG triple. In this work, we focus on a more realistic setting where we aim to generate questions from a KG subgraph and target answers. In addition, most of previous works built on either RNN-based or Transformer-based models to encode a linearized KG sugraph, which totally discards the explicit structure information of a KG subgraph.

In order to address the above challenges, we present a subgraph guided Knowledge Graph Question Generation approach with Graph Neural Networks (GNNs). To this end, we introduce for the first time the Graph2Seq architecture for the task of KG-QG to address the second challenge. We then extend the regular GNN-based encoder to allow processing directed and multi-relational KG subgraphs to solve the first challenge. In addition, we propose a simple yet elegant way to leverage the context information from the answers to effectively handle the third challenge. Extensive experimental results demonstrate the superior performance of our proposed model over the state-of-the-art baselines on two QG benchmarks.

We highlight our main contributions as follows:

- We propose a novel Graph2Seq model for subgraph guided KG-QG. The proposed Graph2Seq model employs bidirectional graph embedding and we design two different GNN encoders to effectively encode KG subgraphs with multi-relational edges.

This chapter has been submitted to: Y. Chen, L. Wu, and M. J. Zaki, "Toward subgraph guided knowledge graph question generation with graph neural networks," in *Proc. 2020 Conf. Empirical Meth. Natural Lang. Process.*

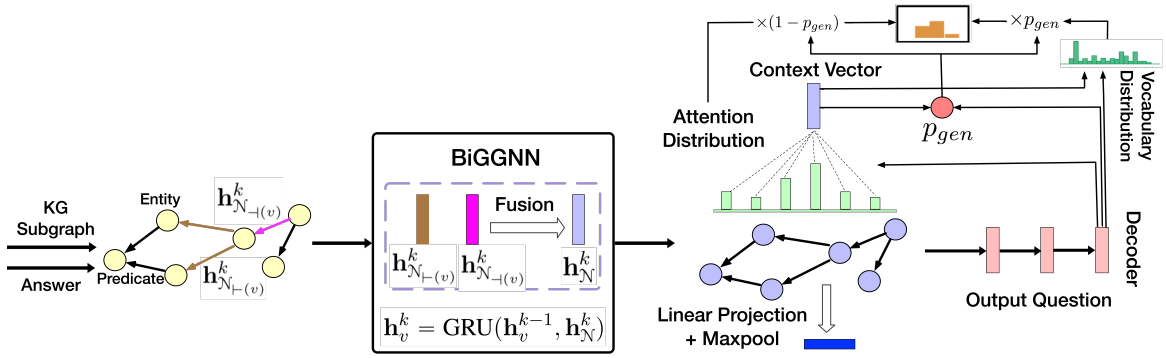

Fig. 5.1: Overall architecture of our proposed model. Best viewed in color.

- We extend the RNN decoder with a novel copying mechanism that allows the entire node attribute to be borrowed from the input KG subgraph when generating questions.
- We investigate two different ways of initializing node/edge embeddings when applying a GNN encoder to process KG subgraphs. In addition, we study the impact of directionality (i.e., bidirectional vs. unidirectional) on GNN encoder.
- Experimental results show that our model improves the state-of-the-art BLEU-4 score from 11.57 to 29.40 and from 25.99 to 59.59 on WQ and PQ benchmarks, respectively. Experiments also show that our QG model can consistently benefit the QA task as a mean of data augmentation.

5.2 Approach: Toward Subgraph Guided Knowledge Graph Question Generation with Graph Neural Networks

In this section, we define the KG-QG task, and then present our novel Graph2Seq model for subgraph guided KG-QG. We first motivate the design, and then present the details of each component as shown in Fig. 5.1.

5.2.1 Problem Formulation

Our focus is on question generation from a KG subgraph, along with potential target answers. We assume that a KG subgraph is a collection of triples (i.e., subject-predicate-object), that can also be represented as a graph $\mathcal{G} = (V, E)$, where $V \in \mathcal{V}$ denotes a set of entities (i.e., subjects or objects) and $E \in \mathcal{E}$ denotes all the predicates connecting these entities. We denote by $\mathcal{V}$ and $\mathcal{E}$ the complete entity set and predicate set of the KG,

respectively. We also assume that all the answers from the target answer set V^a are from the entity set V, which is the normal setting of the task of KBQA [97]. The task of KG-QG is to generate the best NL question consisting of a sequence of word tokens $\hat{Y} = \{y_1, y_2, ..., y_T\}$ which maximizes the conditional likelihood $\hat{Y} = \arg\max_Y P(Y|\mathcal{G}, V^a)$ where T is the length of the question. We focus on the problem setting where we have a set of KG subgraphs (and answers) and target questions pairs, to learn the mapping; existing QG approaches [66], [68], [69] make a similar assumption. Although the three main challenges we have discussed before are based on QG for texts, other QG tasks from other data sources also share some or most of issues when dealing with these tasks. Therefore, our model could be used or adapted to generalize to cope with these tasks as well.

5.2.2 Encoding Layer

Let us denote V as a set of nodes (i.e., entities) $\{v_1, v_2, ..., v_n\}$ in a KG subgraph $\mathcal{G}$, where each node is associated with some attributes such as text or ID. Similarly, let us denote E as a set of edges (i.e., predicates) $\{e_1, e_2, ..., e_m\}$ in $\mathcal{G}$, where each edge has some initial attributes such as text or ID.

5.2.2.1 *Encoding Nodes and Edges*

Before applying the GNN encoder to process a KG subgraph, we need to map nodes and edges to an initial embedding space that encodes their attributes. There are two common ways of encoding nodes and edges in a KG. One solution is based on global KG embeddings that are pretrained on the whole KG by some KG representation learning algorithm such as TransE [278], while the other one is based on pretrained embeddings (e.g., GloVe [273]) of the words making up the textual attributes. In this work, we choose to encode nodes and edges based on word embeddings of their textual attributes in our main model. We posit that it is relatively easier for a model to learn the mapping from the input KG subgraph to the output NL question with both sides based on word embeddings. We empirically compare and analyze the two encoding strategies in our experiments.

In order to encode the nodes and edges in a KG subgraph, we apply two bidirectional LSTMs [172] for nodes (i.e., one for nodes, and one for edges) to encode their associated text. The concatenation of the last forward and backward hidden states of the BiLSTM is used as the initial embeddings for nodes and edges.

5.2.2.2 *Utilizing Target Answers*

In the setting of KBQA [97], [99], it is usually assumed that the answers to a question are entities in a KG subgraph. Since KG-QG is a dual task of KBQA, we assume that utilizing the target answers along with the KG subgraph can help generate more relevant questions. To this end, we apply a simple yet effective strategy where we introduce an additional learnable markup vector associated with each node/edge to indicate whether it is an answer or not.

Therefore, the initial vector representation of a node/edge will be the concatenation of the BiLSTM output and the answer markup vector. We denote $\mathbf{X}^e = \{\mathbf{x}_1^e, \mathbf{x}_2^e, ..., \mathbf{x}_n^e\}$ and $\mathbf{X}^p = \{\mathbf{x}_1^p, \mathbf{x}_2^p, ..., \mathbf{x}_m^p\}$ as the embeddings of the entity nodes and predicate edges, respectively. Both $\mathbf{X}^e$ and $\mathbf{X}^p$ have the same embedding dimension d.

5.2.3 Bidirectional Graph-to-Sequence Generator with Copying Mechanism

RNNs are good at modeling sequential data, however, they cannot handle graph-structured data. One might need to linearize a graph to a sequence so as to apply an RNN-based encoder, which will lose the rich structure information in the graph. [69] proposed to encode a set of triples via a Transformer [108] by removing positional encoding in the original architecture. Even though a Transformer-based encoder might be able to learn the semantic relations among the triples through the all-to-all attention, the explicit graph structure is totally discarded. Recently, GNNs have been used to encode graph-structured data and the resultant models have achieved state-of-the-art performance in many tasks [57], [198], [230], [236]. In this work, we introduce a novel bidirectional GNN-based encoder to encode the KG subgraph, and decode the natural language question via an RNN-based decoder equipped with node-level copying mechanism.

5.2.3.1 *Bidirectional Graph Encoder*

Many existing GNNs [193], [196], [204] were not designed to process directed graphs such as a KG. Even though some GNN variants such as GGSNN [197] and MPNN [195] are able to handle directed graphs via message passing across graphs, they do not model the bidirectional information when aggregating information from neighboring nodes for each node. As a result, messages can only be passed across graphs in a unidirectional way.

In this work, we introduce the Bidirectional Gated Graph Neural Network (BiGGNN)

which extends GGSNN [197] by learning node embeddings from both incoming and outgoing directions in an interleaved fashion when processing a directed graph. A similar bidirectional approach has been exploited in [198], which extended another popular variant of GNNs - GraphSAGE [196]. While their method simply learns the node embeddings of each direction independently and concatenates them at the final step, BiGGNN fuses the intermediate node embeddings from both directions at every iteration.

The embedding $\mathbf{h}_v^0$ for node v is initialized to $\mathbf{x}_v$, that is, a concatenation of the BiLSTM output and the answer markup vector. BiGGNN then performs message passing across the graph for a fixed number of hops, with the same set of network parameters shared at each hop.

At each hop of computation, for every node in the graph, we apply an aggregation function that takes as input a set of incoming (or outgoing) neighboring node vectors and outputs a backward (or forward) aggregation vector. In principle, many order-invariant operators such as max or attention [204] can be employed to aggregate neighborhood information. Here we use a simple average aggregator as follows,

$$\mathbf{h}^k_{\mathcal{N}_{\dashv(v)}} = \text{AVG}(\{\mathbf{h}_v^{k-1}\} \cup \{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}_{\dashv(v)}\}) \tag{5.1a}$$

$$\mathbf{h}^k_{\mathcal{N}_{\vdash(v)}} = \text{AVG}(\{\mathbf{h}_v^{k-1}\} \cup \{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}_{\vdash(v)}\}) \tag{5.1b}$$

where $\mathcal{N}_{\dashv(v)}$ and $\mathcal{N}_{\vdash(v)}$ denote the incoming and outgoing neighbors of v. We then fuse the node embeddings aggregated from both directions,

$$\mathbf{h}^k_{\mathcal{N}_{(v)}} = \text{Fuse}(\mathbf{h}^k_{\mathcal{N}_{\dashv(v)}}, \mathbf{h}^k_{\mathcal{N}_{\vdash(v)}}) \tag{5.2}$$

The fusion function is computed as a gated sum of two information sources,

$$\text{Fuse}(\mathbf{a}, \mathbf{b}) = \mathbf{z} \odot \mathbf{a} + (1 - \mathbf{z}) \odot \mathbf{b} \tag{5.3a}$$

$$\mathbf{z} = \sigma(\mathbf{W}_z[\mathbf{a}; \mathbf{b}; \mathbf{a} \odot \mathbf{b}; \mathbf{a} - \mathbf{b}] + \mathbf{b}_z) \tag{5.3b}$$

where $\odot$ is the component-wise multiplication, σ is a sigmoid function, and $\mathbf{z}$ is a gating vector. The gate helps the model to determine how much of the information needs to be reserved from the two aggregated node embeddings.

Finally, a Gated Recurrent Unit (GRU) [78] is used to update the node embeddings

by incorporating the aggregation information.

$$\mathbf{h}_v^k = \text{GRU}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}_{(v)}}^k) \tag{5.4}$$

After n hops of GNN computation where n is a hyperparameter, we obtain the final state embedding $\mathbf{h}_v^n$ for node v. To compute the graph-level embedding, we first apply a linear projection to the node embeddings, and then apply max-pooling over all node embeddings to get a d-dim vector $\mathbf{h}^{\mathcal{G}}$.

5.2.3.2 *Handling Multi-relational Graphs*

Knowledge graphs are typically heterogeneous networks that contain a large number of edge types. However, many existing GNNs [193], [196], [197], [204] are not directly applicable to multi-relational graphs. In order to model both node and edge information with GNNs, researchers have extended them by either having separate learnable weights for different edge types or having explicit edge embeddings when performing message passing [195], [279]. While the former solution may have severe scalability issues when handling graphs with a large number of edge types, the later one requires major modifications to existing GNN architectures. In this work, we explore two solutions to adapt GNNs to multi-relational graphs as detailed below.

Levi graph transformation. We can directly apply regular GNNs to a multi-relational KG subgraph by converting it to a Levi graph [280]. Specifically, we treat all edges in the original graph as new nodes and add new edges connecting original nodes and new nodes, which results in a bipartite graph. For instance, in a KG subgraph, a triple (Mario_Siciliano, place_of_birth, Rome) will be converted to "Mario_Siciliano → place_of_birth → Rome" where "place_of_birth" becomes a new node, and → indicates a new edge connecting an entity and a predicate. Note that since most KG subgraphs are sparse, the number of newly added nodes (and edges as well) will at most be linear to the number of original nodes.

Gated message passing with edge information. We also extend BiGGNN to explicitly incorporate edge embeddings when conducting message passing, calling the resultant variant

as BiGGNN$_{edge}$. Specifically, we rewrite the node aggregation function Eq. (5.1) as follows,

$$\mathbf{h}^k_{\mathcal{N}_{\dashv(v)}} = \text{AVG}(\{\mathbf{h}^{k-1}_v\} \cup \{f([\mathbf{h}^{k-1}_u; \mathbf{e}_{uv}]), \forall u \in \mathcal{N}_{\dashv(v)}\}) \tag{5.5a}$$

$$\mathbf{h}^k_{\mathcal{N}_{\vdash(v)}} = \text{AVG}(\{\mathbf{h}^{k-1}_v\} \cup \{f([\mathbf{h}^{k-1}_u; \mathbf{e}_{uv}], \forall u \in \mathcal{N}_{\vdash(v)}\}) \tag{5.5b}$$

where f is a nonlinear function (i.e., linear projection + ReLU [281]) applied to the concatenation of $\mathbf{h}^{k-1}_u$ and $\mathbf{e}_{uv}$ which is the embedding of the edge connecting node u and v.

5.2.3.3 *RNN Decoder with Node-level Copying*

We adopt an attention-based [79], [140] LSTM decoder that generates the output sequence one word at a time. The decoder takes the graph-level embedding $\mathbf{h}^{\mathcal{G}}$ followed by two separate fully-connected layers as initial hidden states (i.e., $\mathbf{c}_0$ and $\mathbf{s}_0$) and the node embeddings $\{\mathbf{h}^n_v, \forall v \in \mathcal{G}\}$ as the attention memory. The particular attention mechanism used in our decoder closely follows [41]. Basically, at each decoding step t, an attention mechanism learns to attend to the most relevant nodes in the input graph, and computes a context vector $\mathbf{h}^*_t$ based on the current decoding state $\mathbf{s}_t$ and the attention memory.

We hypothesize that when generating NL questions from a KG subgraph, it is very likely to directly mention (i.e., copy) entity names that are from the input KG subgraph even without rephrasing them. When augmented with copying mechanism [154], [155], most RNN decoders are typically allowed to copy words from the input sequence. We extend the regular word-level copying mechanism to the node-level copying mechanism that allows copying node attributes (i.e., node text) from the input graph. At each decoding step, the generation probability $p_{\text{gen}} \in [0, 1]$ is calculated from the context vector $\mathbf{h}^*_t$, the decoder state $\mathbf{s}_t$ and the decoder input y_{t-1}. Next, p_{gen} is used as a soft switch to choose between generating a word from the vocabulary or copying a node attribute from the input graph. We dynamically maintain an extended vocabulary which is the union of the usual vocabulary and all node names appearing in a batch of source examples (i.e., KG subgraphs).

5.2.4 Training and Testing

As customary for training sequential models, we minimize the following cross-entropy loss,

$$\mathcal{L}_{lm} = \sum_t -\log P(y^*_t | X, y^*_{<t}) \tag{5.6}$$

where y_t^* is the word at the t-th position of the gold output sequence. Scheduled teacher forcing [282] is adopted to alleviate the exposure bias problem. During the testing phase, beam search is applied to generate the output.

Besides training our proposed model with the regular cross-entropy loss, we also explore minimizing a hybrid objective combining both the cross-entropy loss and Reinforcement Learning (RL) [111] loss that is defined based on evaluation metrics, as detailed next.

Hybrid evaluator. Most prior works on QG employ cross-entropy based training objective, which is also a de facto choice for training sequential models in many other NLP tasks. However, cross-entropy based training strategy has some known limitations including exposure bias and evaluation discrepancy between training and testing [81], [109], [110]. That is to say, in the training phase, a model has access to the ground-truth previous token when decoding and is optimized toward cross-entropy loss, while in the testing phase, no ground-truth previous token is provided and cross-entropy loss is not used for evaluation. To tackle these issues, we introduce a hybrid objective function combining both cross-entropy loss and Reinforcement Learning (RL) [111] loss for training our Graph2Seq model.

Specifically, we introduce a two-stage training strategy as follows. In the first stage, the regular cross-entropy loss is used,

$$\mathcal{L}_{lm} = \sum_t -\log P(y_t^*|X, y_{<t}^*) \tag{5.7}$$

where y_t^* is the word at the t-th position of the ground-truth output sequence. Scheduled teacher forcing [282] is adopted to alleviate the exposure bias problem. In the second stage, we further fine-tune the model by optimizing a mixed objective function combining both cross-entropy loss and RL loss, defined as,

$$\mathcal{L} = \gamma\mathcal{L}_{rl} + (1-\gamma)\mathcal{L}_{lm} \tag{5.8}$$

where γ is a scaling factor controling the trade-off between the two losses. During the testing phase, beam search is applied to generate the output.

While our architecture is agnostic to the specific RL algorithm, in this work, we employ an efficient yet effective RL approach called self-critical sequence training (SCST) [283] to directly optimize the discrete evaluation metrics. At each training iteration, the RL loss is

defined by comparing the reward of the sampled output Y^s with the reward of the baseline output $\hat{Y}$,

$$\mathcal{L}_{rl} = (r(\hat{Y}) - r(Y^s)) \sum_t \log P(y_t^s | X, y_{<t}^s) \tag{5.9}$$

where Y^s is produced by multinomial sampling, that is, each word y_t^s is sampled according to the likelihood $P(y_t|X, y_{<t})$ predicted by the generator, and $\hat{Y}$ is obtained by greedy search, that is, by maximizing the output probability distribution at each decoding step. As we can see, minimizing the above loss is equivalent to maximizing the likelihood of some sampled output that has a higher reward than the corresponding baseline output.

One of the key factors for RL is to pick the proper reward function. We define $r(Y)$ as the reward of an output sequence Y, computed by comparing it to the corresponding ground-truth sequence Y^* with some reward metric which is a combination of our evaluation metrics (e.g., BLEU-4, ROUGE-L, etc.). This lets us directly optimize the model towards the evaluation metrics.

5.3 Experiments

In this section, we conduct extensive experiments to evaluate the effectiveness of our proposed model for the QG task. We also conduct experiments to examine whether our QG model can help the QA task by providing more training data. Besides, we want to examine whether the introduced GNN-based encoder works better than an RNN-based or Transformer-based encoder when encoding a KG subgraph for the QG task. In addition, we explore and analyze two different ways of handling multi-relational graphs with GNNs. Moreover, we empirically compare two different ways of initializing node and edge embeddings before feeding them into a GNN-based encoder. An experimental comparison between bidirectional GNN-based encoder and unidirectional GNN-based encoder is also provided. The implementation of our model is publicly available at `https://github.com/hugochan/Graph2Seq-for-KGQG`.

5.3.1 Baseline Methods

We compare our model against the following baselines: i) L2A [73], ii) Transformer [108], and iii) MHQG+AE [69]. Detailed descriptions of the baselines are provided next.

L2A is a LSTM-based Seq2Seq model equipped with attention mechanism, which takes as

input a linearized KG subgraph. It was included in [69] as a baseline. The results of L2A reported here are taken from [69].

Transformer We also include a Transformer-based encoder-decoder model [284] with copying mechanism that takes as input a linearized KG subgraph, i.e., a sequence of triples where each triple is represented as a sequence of tokens containing the subject name, predicate name and object name.

MHQG+AE employs a Transformer-based encoder to encode a KG subgraph (i.e., a set of triples), and generates an output question with a Transformer-based decoder. Unlike the above Transformer baseline, the MHQG+AE model encodes the triple set contained in a KG subgraph by removing the positional encoding in a regular Transformer architecture, and their triple embeddings are based on the KG embeddings pretrained by a knowledge-base representation learning framework called TransE [278]. To the best of our knowledge, MHQG+AE was probably the first NN-based model that focused on QG from a KG subgraph.

5.3.2 Data and Metrics

Following [69], we used WebQuestions (WQ) and PathQuestions (PQ) as our benchmarks where both of them use Freebase [12] as the underlying KG. The WQ dataset combines examples from WebQuestionsSP [285] and ComplexWebQuestions [286] where both of them are question answering datasets that contain natural language questions, corresponding SPARQL queries and answer entities. For each instance in WQ, in order to construct the KG subgraph, [69] converted its SPARQL query to return a subgraph instead of the answer entity, by changing it from a SELECT query to a CONSTRUCT query. The WQ dataset [69] contains 18,989/2,000/2,000 (train/development/test) examples. The PQ dataset [287] is similar to WQ except that the KG subgraph in PQ is a path between two entities that span two or three hops. The PQ dataset contains 9,793/1,000/1,000 (train/development/test) examples. Brief statistics of the two datasets are provided in Table 5.1.

Following previous works, we use BLEU-4 [113], METEOR [114] and ROUGE-L [115] as automatic evaluation metrics. Initially, BLEU-4 and METEOR were designed for evaluating machine translation systems and ROUGE-L was designed for evaluating text summarization systems.

`https://github.com/liyuanfang/mhqg`

Table 5.1: Data statistics. The min/max/avg statistics are reported on KG triples and queries.

Data	# examples	# entities	# predicates	# triples	query length
WQ	22,989	25,703	672	2/99/5.8	5/36/15
PQ	9,731	7,250	378	2/3/2.7	8/25/14

We also conducted a small-scale (i.e., 50 random examples per system) human evaluation study on the WQ test set. We asked 6 human evaluators to give feedback on the quality of questions generated by a set of anonymized competing systems. In each example, given a KG subgraph, target answers and an anonymized system output, they were asked to rate the quality of the output by answering the following three questions: i) is this generated question syntactically correct? ii) is this generated question semantically correct? and iii) is this generated question relevant to the KG subgraph and target answers? For each evaluation question, the rating scale is from 1 to 5 where a higher score means better quality (i.e., 1: Poor, 2: Marginal, 3: Acceptable, 4: Good, 5: Excellent). Responses from all evaluators were collected and averaged.

5.3.3 Model Settings

We keep and fix the 300-dim GloVe [273] vectors for those words that occur more than twice in the training set. The dimensions of answer markup embeddings are set to 32 and 24 for WQ and PQ, respectively. We set the hidden state size of BiLSTM to 150 so that the concatenated state size for both directions is 300. The size of all other hidden layers is set to 300. We apply a variational dropout [288] rate of 0.4 after word embedding layers and 0.3 after RNN layers. The label smoothing ratio is set to 0.2. The number of GNN hops is set to 4. During training, in each epoch, we set the initial teacher forcing probability to 0.8 and exponentially increase it to $0.8 * 0.9999^i$ where i is the training step. In addition, partial teacher forcing is adopted, which means that when generating a sequence, some steps can be teacher forced and some not. We set γ in the mixed loss function to 0.99. We use Adam [274] as the optimizer. The learning rate is set to 0.001 in the pretraining stage. In the fine-tuning stage, we set the learning rate to 0.00001 and 0.00002 for WQ and PQ, respectively. We reduce the learning rate by a factor of 0.5 if the validation BLEU-4 score stops improving for three epochs. We stop the training when no improvement is seen for 10 epochs. We clip

Table 5.2: Automatic evaluation results on the WQ and the PQ test sets.

Method	WQ			PQ		
	BLEU-4	METEOR	ROUGE-L	BLEU-4	METEOR	ROUGE-L
L2A	6.01	25.24	26.95	17.00	19.72	50.38
Transformer	8.94	13.79	32.63	56.43	43.45	73.64
MHQG+AE	11.57	29.69	35.53	25.99	33.16	58.94
G2S+AE	**29.45**	30.96	**55.45**	**61.48**	44.57	**77.72**
G2S$_{edge}$ +AE	29.40	**31.12**	55.23	59.59	**44.70**	75.20

the gradient at length 10. The batch size is set to 30. The beam search width is set to 5. All hyperparameters are tuned on the development set. Experiments were conducted on a machine which has an Intel i7-2700K CPU and an Nvidia Titan Xp GPU with 16GB RAM.

5.3.4 Experimental Results

Table 5.3: Human evaluation results (± standard deviation) on the WQ test set.

Method	Syntactic	Semantic	Relevant	Overall
Transformer	**4.53 (0.18)**	**4.58 (0.22)**	2.65 (0.57)	3.92 (0.24)
G2S+AE	4.18 (0.30)	4.30 (0.27)	4.26 (0.34)	4.25 (0.26)
Ground-truth	4.30 (0.15)	4.50 (0.18)	**4.32 (0.32)**	**4.38 (0.19)**

Table 5.2 shows the evaluation results comparing our proposed models against other state-of-the-art baseline methods on WQ and PQ test sets. As we can see, our models outperform all baseline methods by a large margin on both benchmarks. Besides, we can clearly see the advantages of GNN-based encoders for modeling KG subgraphs, by comparing our model with RNN-based (i.e., L2A) and Transformer-based (i.e., Transformer, MHQG+AE) baselines. Compared to our Graph2Seq model, both RNN-based and Transformer-based baselines ignore the explicit graph structure of a KG subgraph, which leads to degraded performance. Interestingly, the Transformer baseline performs reasonably well on PQ, but dramatically fails on WQ. We speculate this is because PQ is more friendly to sequential models such as Transformer as the KG subgraph in PQ is more like path-structure while the one in WQ is more like tree-structure.

We also compare two variants of our model (i.e., G2S vs. G2S$_{edge}$) for handling multi-relational graphs. As shown in Table 5.2, directly applying the BiGGNN encoder to a Levi

Table 5.4: Ablation study on the WQ and the PQ test sets.

Method	WQ			PQ		
	BLEU-4	METEOR	ROUGE-L	BLEU-4	METEOR	ROUGE-L
G2S+AE	**29.45**	**30.96**	**55.45**	**61.48**	**44.57**	**77.72**
G2S	28.43	30.13	54.44	60.68	44.07	75.94
G2S w/o copy	22.95	26.99	51.05	57.10	42.66	74.29

graph which is converted from a KG subgraph works quite well. The proposed BiGGNN$_{edge}$ model can directly handle multi-relational graphs without modifying the input graph. However, it performs slightly worse than the Levi graph solution. We can improve the modeling power of BiGGNN$_{edge}$ by updating edge embeddings in the message passing process and attending to edges in the attention mechanism. We leave these extensions as future work.

Human evaluation and error analysis. We conduct a human evaluation study to assess the quality of the questions generated by our model, the Transformer baseline, and the ground-truth data in terms of syntax, semantics and relevance metrics. In addition, an overall score is computed for each example by taking the average of the three scores. As shown in Table 5.3, overall, we can see that our model achieves good results even compared to the ground-truth, and outperforms the Transformer baseline. Interestingly, we observe that the Transformer baseline gets high syntactic and semantic scores, but very poor relevant scores. After manually examining some generated questions, we noticed that it generates many fluent and meaningful questions that are by no means relevant to the given KG subgraph. However, our model is able to generate more relevant questions possibly by better capturing the KG semantics and the answer information. Our error analysis shows that main errors of our model occur in repeated words and grammatical errors in generated questions.

5.3.5 Ablation Study

As shown in Table 5.4, we perform an ablation study to assess the performance impacts of different model components. First of all, the node-level copying mechanism contributes a lot to the overall model performance. By turning it off, we observe significant performance drops on both benchmarks. This verifies our assumption that when generating questions from a KG subgraph, one usually directly copies named entities from the input KG subgraph to the output question. Besides, the answer information is also important for generating relevant questions. Even with the simple answer markup technique, we can see the performance boost

on both benchmarks.

5.3.6 Model Analysis

Effect of node/edge embedding initialization. We empirically compare two different ways of initializing node/edge embeddings when applying the Graph2Seq model. As shown in Table 5.5, encoding nodes and edges based on word embeddings of their textual attributes works better than based on their KG embeddings. This might be because it is difficult for a NN-based model to learn the gap between KG embeddings on the encoder side and word embeddings on the decoder side. With the word embedding-based encoding strategy, it is relatively easier for a model to learn the mapping from the input KG subgraph to the output NL question. It also seems that modeling local dependency within the subgraph without utilizing the global KG information is enough for generating meaningful questions from a KG subgraph.

Table 5.5: Effect of node/edge initial embeddings on the WQ test set.

Method	BLEU-4	METEOR	ROUGE-L
w/ word emb.	**28.43**	**30.13**	**54.44**
w/ KG emb.	22.80	25.85	48.93

Table 5.6: Impact of directionality for G2S+AE on the PQ test set.

Method	BLEU-4	METEOR	ROUGE-L
Bidirectional	**61.48**	**44.57**	**77.72**
Forward	59.59	42.72	75.82
Backward	59.12	42.66	75.03

Table 5.7: Results of RL-based G2S+AE on the WQ test set.

Method	BLEU-4	METEOR	ROUGE-L
G2S+AE	29.45	30.96	55.45
G2S+AE+RL	**29.80**	**31.29**	**55.51**

Impact of directionality on GNN encoder. As show in Table 5.6, we compare the performance of bidirectional Graph2Seq with unidirectional Graph2Seq. We observe that performing unidirectional message passing degrades the performance.

Table 5.8: Results of RL-based G2S+AE on the PQ test set.

Method	BLEU-4	METEOR	ROUGE-L
G2S+AE	**61.48**	**44.57**	**77.72**
G2S+AE+RL	59.21	44.47	77.35

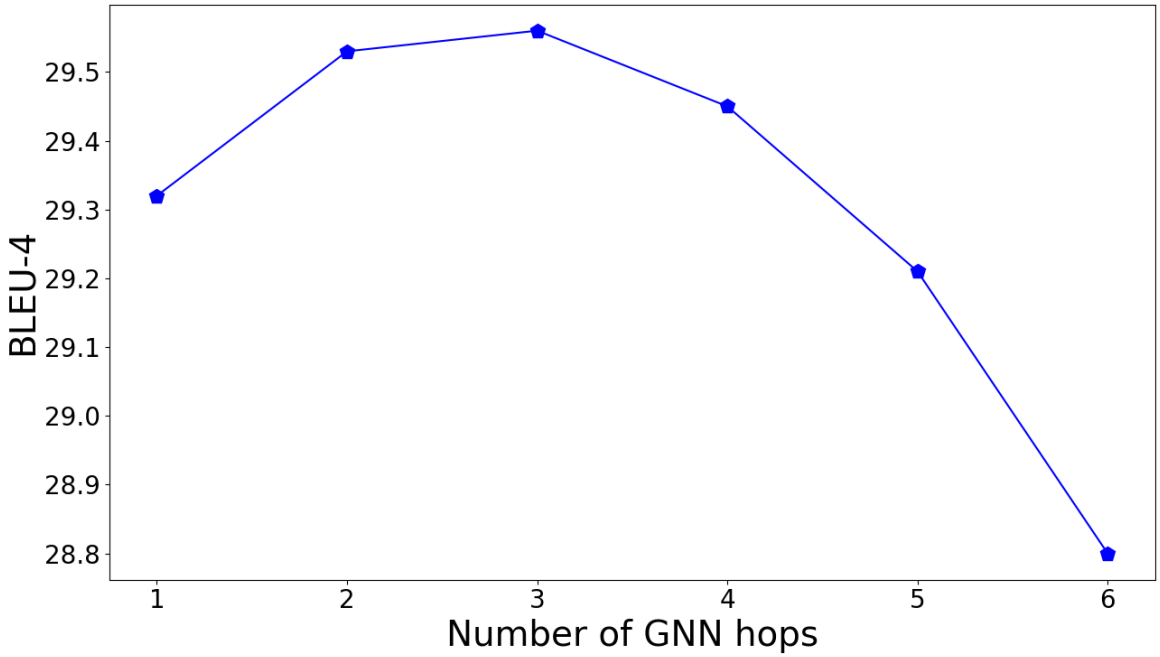

Fig. 5.2: Effect of the number of GNN hops for G2S+AE on the PQ test set.

Results on training the model with a hybrid objective. Table 5.7 and Table 5.8 show the results of training our proposed G2S+AE model with a hybrid objective combining both cross-entropy loss and RL loss. While the RL-based training strategy boosts the model performance on WQ, it does not help the model training on PQ.

Effect of the number of GNN hops. Fig. 5.2 shows the impact of the number of GNN hops when applying a GNN-based encoder to encode the KG subgraph in WQ. It indicates that increasing the number of GNN hops can boost the model performance until some optimal value.

5.3.7 Case Study

As shown in Table 5.9, we conduct a case study to examine the quality of generated questions using different ablated systems. First of all, by initializing node/edge embeddings with KG embeddings, the model fails to generate reasonable questions. Besides, with the

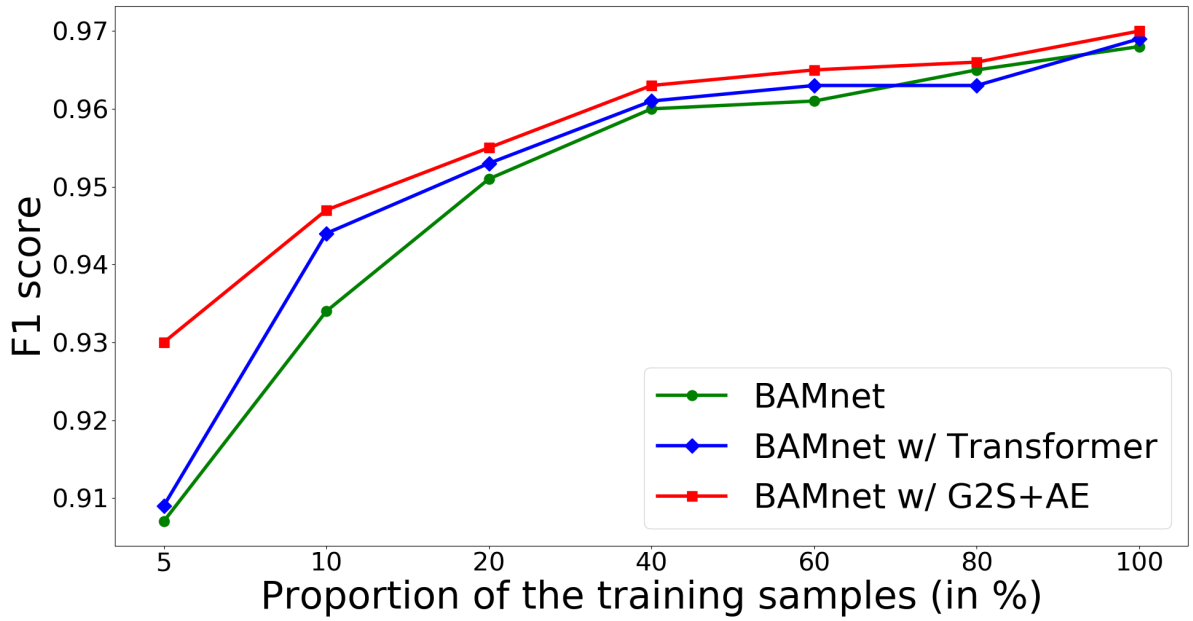

Fig. 5.3: Performance of QG-driven KBQA method under different proportions of training data.

node-level copying mechanism, the model is able to directly copy the entity name "giza necropolis" into the output question. Last, incorporating the answer information helps generate more relevant and specific questions.

Table 5.9: Generated questions on the WQ test set. Target answers are underlined. For the sake of brevity, we only display the lowest level of the predicate hierarchy.

KG subgraph: (Egypt, administrative_divisions, Cairo), (Giza Necropolis, containedby, Egypt)
Gold: what country has the city of cairo and is home of giza necropolis ?
G2S w/ KG emb.: what country that contains cairo has cairo as its province ?
G2S w/o copy: where is the giza giza located in that has cairo ?
G2S: where is the giza necropolis located in that contains cairo ?
G2S+AE: what country that contains cairo is the location of giza necropolis ?

5.3.8 QG-Driven Data Augmentation for QA

One of the most important applications of QG is to generate more training data for QA tasks. In this section, we use our proposed QG model to generate more questions for training KBQA methods. We use WQ as our KBQA benchmark, and randomly split it to 40%/20%/40% (train/dev/test) examples. As for the KBQA baseline, we use the state-of-

the-art KBQA model called BAMnet [97] which directly retrieves answers from a KG by mapping questions and candidate answers into a joint embedding space. In order to examine the effect of QG-driven data augmentation on the KBQA task, we compare the BAMnet baseline with its two data augmentation variants, namely, BAMnet w/ Transformer and BAMnet w/ G2S+AE. More specifically, the BAMnet baseline is trained only on the part (i.e., x% of the whole training data) where gold questions are available, while the other two variants are trained on the combination of the gold questions and the questions automatically generated by two QG models. Note that given x% training data, we randomly split it to 80%/20% (train/dev) for training a QG model.

We gradually increase the proportion (i.e., x%) of the training data used for training KBQA models, and report the F1 score performance of the above three KBQA model variants. Here F1 score measures the overlap between the predicted and ground-truth answer set. Fig. 5.3 shows the results on improving the BAMnet baseline with automatically generated questions. Interestingly, both QG models consistently help improve the KBQA performance when varying x% training data, and the performance boost is the most significant when training data is scarce. Notably, our G2S+AE model consistently outperforms the Transformer model in improving the KBQA performance.

5.4 Conclusion and Future Work

In this paper, we introduced a novel bidirectional Graph2Seq model for the KG-QG task. A novel node-level copying mechanism was proposed to allow directly copying node attributes from the KG subgraph to the output question. We explored different ways of initializing node/edge embeddings and handling multi-relational graphs. Our model outperforms existing methods by a significant margin on both WQ and PQ benchmarks.

Future directions include exploring effective ways of initializing node/edge embeddings and utilizing answer information.

CHAPTER 6
NATURAL QUESTION GENERATION FROM TEXT

6.1 Overview

Natural question generation (QG) aims to generate questions from a text passage and an answer. Previous works on QG either (i) ignore the rich structure information hidden in text, (ii) solely rely on cross-entropy loss that leads to issues like exposure bias and inconsistency between train/test measurement, or (iii) fail to fully exploit the answer information.

To address these limitations, in this work, we present a novel reinforcement learning (RL) [111] based generator-evaluator architecture that aims to: i) make full use of rich hidden structure information beyond the simple word sequence; ii) generate syntactically and semantically valid text while maintaining the consistency of train/test measurement; iii) model explicitly the global interactions of semantic relationships between passage and answer at both word-level and contextual-level.

In particular, to achieve the first goal, we explore two different means to either construct a syntax-based static graph or a semantics-aware dynamic graph from the text sequence, as well as its rich hidden structure information. Then, we design a graph-to-sequence (Graph2Seq) [198], [227], [230]–[233] model based generator that encodes the graph representation of a text passage and decodes a question sequence using a Recurrent Neural Network (RNN) [172]. Our Graph2Seq model is based on a novel bidirectional gated graph neural network, which extends the gated graph neural network [197] by considering both incoming and outgoing edges, and fusing them during the graph embedding learning. To achieve the second goal, we design a hybrid evaluator which is trained by optimizing a mixed objective function that combines both cross-entropy and RL loss. We use not only discrete evaluation metrics like BLEU, but also semantic metrics like word mover's distance [289] to encourage both syntactically and semantically valid text generation. To achieve the third goal, we propose a novel Deep Alignment Network (DAN) for effectively incorporating answer information into the passage at multiple granularity levels.

Our main contributions are as follows:

This chapter previously appeared as: Y. Chen, L. Wu, and M. J. Zaki, "Reinforcement learning based graph-to-sequence model for natural question generation," in *Proc. 8th Int. Conf. Learn. Representations*, Apr. 26-30, 2020. [Online]. Available: https://openreview.net/forum?id=HygnDhEtvr

- We propose a novel RL-based Graph2Seq model for natural question generation. To the best of our knowledge, we are the first to introduce the Graph2Seq architecture for the QG task.
- We design a novel Bidirectional Gated Graph Neural Network (BiGGNN) to process directed passage graphs.
- We design an effective Deep Alignment Network (DAN) for incorporating answer information into the passage with multiple granularity levels.
- We design a two-stage training strategy to train the proposed model with both cross-entropy loss and RL loss.
- We explore both static and dynamic ways of constructing graph from text and are the first to systematically investigate their performance impacts on a GNN encoder.
- The proposed model is end-to-end trainable, achieves new state-of-the-art scores, and outperforms existing methods by a significant margin on the standard SQuAD benchmark for QG. Our human evaluation study also corroborates that the questions generated by our model are more natural (semantically and syntactically) compared to other baselines.

6.2 Approach: RL-based Graph2Seq Model for Natural Question Generation

In this section, we define the question generation task, and then present our RL-based Graph2Seq model for natural question generation. We first motivate the design, and then present the details of each component as shown in Fig. 6.1.

6.2.1 Problem Formulation

The goal of question generation is to generate natural language questions based on a given form of data, such as knowledge base triples or tables [61], sentences [73], [75], or images [57], where the generated questions need to be answerable from the input data. In this work, we focus on QG from a given text passage, along with a target answer.

We assume that a text passage is a collection of word tokens $X^p = \{x_1^p, x_2^p, ..., x_N^p\}$, and a target answer is also a collection of word tokens $X^a = \{x_1^a, x_2^a, ..., x_L^a\}$. The task of

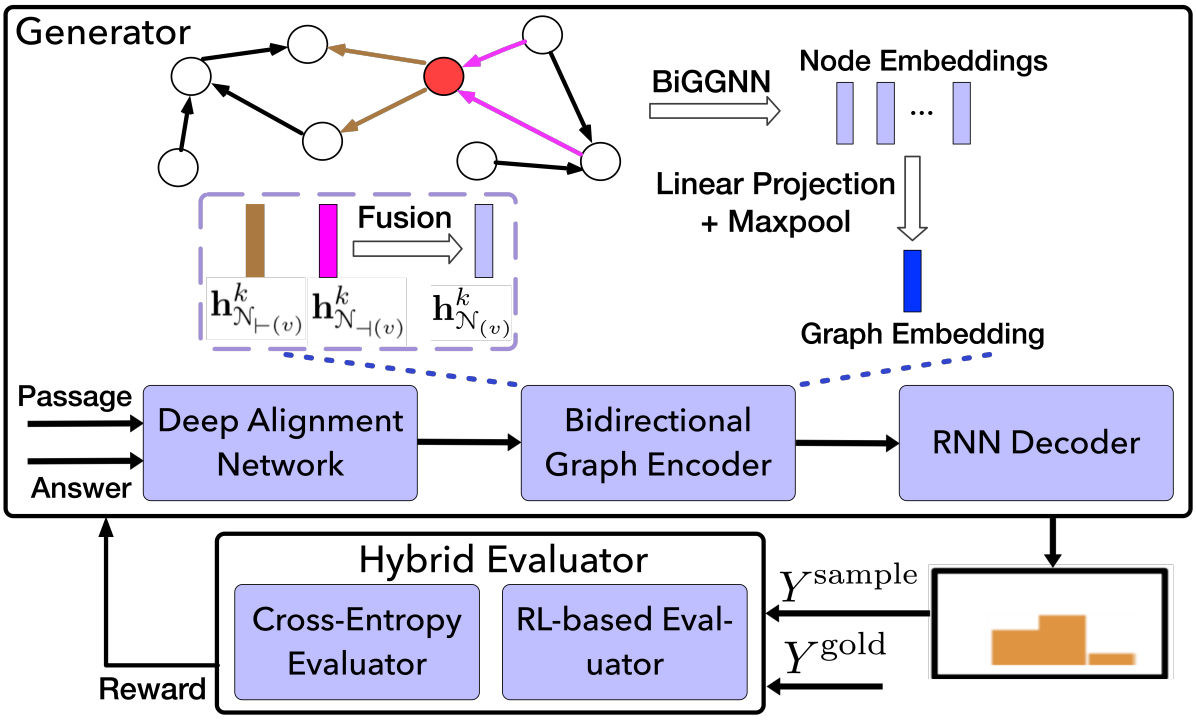

Fig. 6.1: Overall architecture of the proposed model. Best viewed in color.

natural question generation is to generate the best natural language question consisting of a sequence of word tokens $\hat{Y} = \{y_1, y_2, ..., y_T\}$ which maximizes the conditional likelihood $\hat{Y} = \arg\max_Y P(Y|X^p, X^a)$. Here N, L, and T are the lengths of the passage, answer and question, respectively. We focus on the problem setting where we have a set of passage (and answers) and target questions pairs, to learn the mapping; existing QG approaches [73], [75], [87], [90] make a similar assumption.

6.2.2 Deep Alignment Network

Answer information is crucial for generating relevant and high quality questions from a passage. Unlike previous methods that neglect potential semantic relations between passage and answer words, we explicitly model the global interactions among them in the embedding space. To this end, we propose a novel Deep Alignment Network (DAN) component for effectively incorporating answer information into the passage with multiple granularity levels. Specifically, we perform attention-based soft-alignment at the word-level, as well as at the contextual-level, so that multiple levels of alignments can help learn hierarchical representations.

Let $\mathbf{X}^p \in \mathbb{R}^{F \times N}$ and $\widetilde{\mathbf{X}}^p \in \mathbb{R}^{\widetilde{F}_p \times N}$ denote two embeddings associated with passage

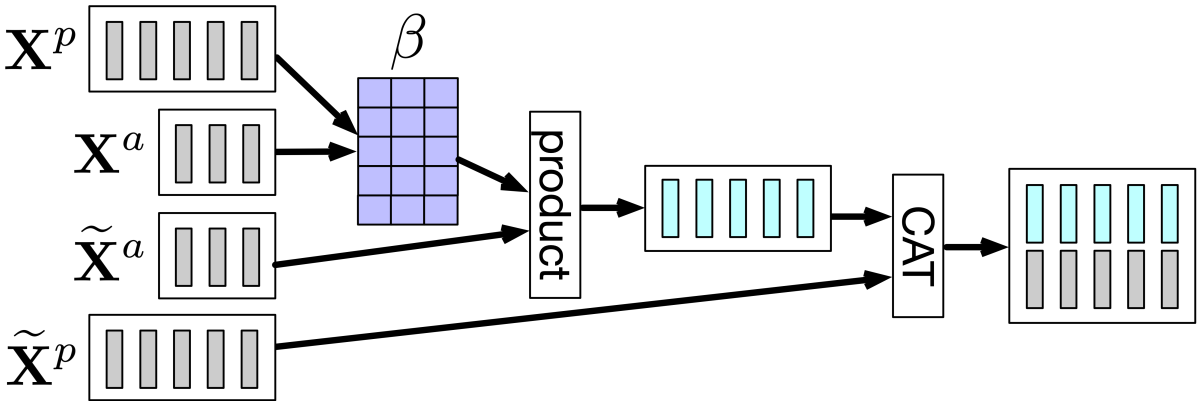

Fig. 6.2: The attention-based soft-alignment mechanism.

text. Similarly, let $\mathbf{X}^a \in \mathbb{R}^{F \times L}$ and $\widetilde{\mathbf{X}}^a \in \mathbb{R}^{\widetilde{F}_a \times L}$ denote two embeddings associated with answer text. Conceptually, as shown in Fig. 6.2, the soft-alignment mechanism consists of three steps: i) compute the attention score $\beta_{i,j}$ for each pair of passage word x_i^p and answer word x_j^a: ii) multiply the attention matrix $\boldsymbol{\beta}$ with the answer embeddings $\widetilde{\mathbf{X}}^a$ to obtain the aligned answer embeddings $\mathbf{H}^p$ for the passage; iii) concatenate the resulting aligned answer embeddings $\mathbf{H}^p$ with the passage embeddings $\widetilde{\mathbf{X}}^p$ to get the final passage embeddings $\widetilde{\mathbf{H}}^p \in \mathbb{R}^{(\widetilde{F}_p + \widetilde{F}_a) \times N}$. Notably, the passage (or answer) embeddings for computing the attention matrix can be same with or different from the passage (or answer) embeddings for computing the final embeddings, as we will discuss below.

Formally, we define our soft-alignment function as following:

$$\widetilde{\mathbf{H}}^p = \text{Align}(\mathbf{X}^p, \mathbf{X}^a, \widetilde{\mathbf{X}}^p, \widetilde{\mathbf{X}}^a) \tag{6.1a}$$

$$= \text{CAT}(\widetilde{\mathbf{X}}^p; \mathbf{H}^p) \tag{6.1b}$$

$$= \text{CAT}(\widetilde{\mathbf{X}}^p; \widetilde{\mathbf{X}}^a \boldsymbol{\beta}^T) \tag{6.1c}$$

where the matrix $\widetilde{\mathbf{H}}^p$ is the final passage embedding, the function CAT is a simple concatenation operation, and $\boldsymbol{\beta}$ is a $N \times L$ attention score matrix, computed by

$$\boldsymbol{\beta} \propto \exp\Big(\text{ReLU}(\mathbf{W}\mathbf{X}^p)^T \text{ReLU}(\mathbf{W}\mathbf{X}^a)\Big) \tag{6.2}$$

where $\mathbf{W} \in \mathbb{R}^{d \times F}$ is a trainable weight matrix, with d being the hidden state size and ReLU is the rectified linear unit [281]. After introducing the general soft-alignment mechanism, we next introduce how we do soft-alignment at both word-level and contextual-level.

6.2.2.1 *Word-level Alignment*

In the word-level alignment stage, we first perform a soft-alignment between the passage and the answer based only on their pretrained GloVe embeddings and compute the final passage embeddings by $\widetilde{\mathbf{H}}^p = \text{Align}(\mathbf{G}^p, \mathbf{G}^a, [\mathbf{G}^p; \mathbf{B}^p; \mathbf{L}^p], \mathbf{G}^a)$, where $\mathbf{G}^p$, $\mathbf{B}^p$, and $\mathbf{L}^p$ are the corresponding GloVe embedding [273], BERT embedding [123], and linguistic feature (i.e., case, NER and POS) embedding of the passage text, respectively. Then a bidirectional LSTM [172] is applied to the final passage embeddings $\widetilde{\mathbf{H}}^p = \{\widetilde{\mathbf{h}}_i^p\}_{i=1}^N$ to obtain contextualized passage embeddings $\overline{\mathbf{H}}^p \in \mathbb{R}^{\overline{F} \times N}$.

On the other hand, for the answer text $\mathbf{X}^a$, we simply concatenate its GloVe embedding $\mathbf{G}^a$ and its BERT embedding $\mathbf{B}^a$ to obtain its word embedding matrix $\mathbf{H}^a \in \mathbb{R}^{d' \times L}$. Another BiLSTM is then applied to the concatenated answer embedding sequence to obtain the contextualized answer embeddings $\overline{\mathbf{H}}^a \in \mathbb{R}^{\overline{F} \times L}$.

6.2.2.2 *Contextual-level Alignment*

In the contextual-level alignment stage, we perform another soft-alignment based on the contextualized passage and answer embeddings. Similarly, we compute the aligned answer embedding, and concatenate it with the contextualized passage embedding to obtain the final passage embedding matrix $\text{Align}([\mathbf{G}^p; \mathbf{B}^p; \overline{\mathbf{H}}^p], [\mathbf{G}^a; \mathbf{B}^a; \overline{\mathbf{H}}^a], \overline{\mathbf{H}}^p, \overline{\mathbf{H}}^a)$. Finally, we apply another BiLSTM to the above concatenated embedding to get a $\overline{F} \times N$ passage embedding matrix $\mathbf{X}$.

6.2.3 Bidiectional Graph-to-Sequence Generator

While RNNs are good at capturing local dependencies among consecutive words in text, GNNs have been shown to better utilize the rich hidden text structure information such as syntactic parsing [232] or semantic parsing [233], and can model the global interactions (relations) among sequence words to further improve the representations. Therefore, unlike most of the existing methods that rely on RNNs to encode the input passage, we first construct a passage graph $\mathcal{G}$ from text where each passage word is treated as a graph node, and then employ a novel Graph2Seq model to encode the passage graph (and answer), and to decode the question sequence.

6.2.3.1 *Passage Graph Construction*

Existing GNNs assume a graph structured input and directly consume it for computing the corresponding node embeddings. However, we need to construct a graph from the text. Although there are early attempts on constructing a graph from a sentence [232], there is no clear answer as to the best way of representing text as a graph. We explore both static and dynamic graph construction approaches, and systematically investigate the performance differences between these two methods in the experimental section.

Syntax-based static graph construction. We construct a directed and unweighted passage graph based on dependency parsing. For each sentence in a passage, we first get its dependency parse tree. We then connect neighboring dependency parse trees by connecting those nodes that are at a sentence boundary and next to each other in text.

Semantics-aware dynamic graph construction. We dynamically build a directed and weighted graph to model semantic relationships among passage words. We make the process of building such a graph depend on not only the passage, but also on the answer. The graph construction procedure consists of three steps: i) we compute a dense adjacency matrix $\mathbf{A}$ for the passage graph by applying self-attention to the word-level passage embeddings $\widetilde{\mathbf{H}}^p$, ii) considering that a fully connected passage graph is not only computationally expensive but also might introduce noise (i.e., unimportant edges), a kNN-style graph sparsification strategy [98] is adopted to obtain a sparse adjacency matrix $\bar{\mathbf{A}}$, where we only keep the K nearest neighbors (including itself) as well as the associated attention scores (i.e., the remaining attentions scores are masked off) for each node; and iii) inspired by BiLSTM over LSTM, we also compute two normalized adjacency matrices $\mathbf{A}^{\dashv}$ and $\mathbf{A}^{\vdash}$ according to their incoming and outgoing directions, by applying softmax operation on the resulting sparse adjacency matrix $\bar{\mathbf{A}}$ and its transpose, respectively.

$$\mathbf{A} = \text{ReLU}(\mathbf{U}\widetilde{\mathbf{H}}^p)^T \ \text{ReLU}(\mathbf{U}\widetilde{\mathbf{H}}^p) \tag{6.3a}$$

$$\bar{\mathbf{A}} = \text{kNN}(\mathbf{A}) \tag{6.3b}$$

$$\mathbf{A}^{\dashv}, \mathbf{A}^{\vdash} = \text{softmax}(\{\bar{\mathbf{A}}, \bar{\mathbf{A}}^T\}) \tag{6.3c}$$

where $\mathbf{U}$ is a $d \times (\widetilde{F}_p + \widetilde{F}_a)$ trainable weight matrix. Note that the supervision signal is able to back-propagate through the graph sparsification operation as the K nearest attention scores are kept.

6.2.3.2 *Bidirectional Gated Graph Neural Networks*

To effectively learn the graph embeddings from the constructed text graph, we propose a novel Bidirectional Gated Graph Neural Network (BiGGNN) which extends Gated Graph Sequence Neural Networks [197] by learning node embeddings from both incoming and outgoing edges in an interleaved fashion when processing the directed passage graph. Similar idea has also been exploited in [198], which extended another popular variant of GNNs - GraphSAGE [196]. However, one of key difference between our BiGGNN and their bidirectional GraphSAGE is that we fuse the intermediate node embeddings from both incoming and outgoing directions in every iteration, whereas their model simply learns the node embeddings of each direction independently and concatenates them in the final step.

In BiGGNN, node embeddings are initialized to the passage embeddings $\mathbf{X}$ returned by DAN. The same set of network parameters are shared at every hop of computation. At each computation hop, for every node in the graph, we apply an aggregation function which takes as input a set of incoming (or outgoing) neighboring node vectors and outputs a backward (or forward) aggregation vector. For the syntax-based static graph, we use a mean aggregator for simplicity although other operators such as max or attention [204] could also be employed,

$$\mathbf{h}^k_{\mathcal{N}_{\dashv(v)}} = \text{MEAN}(\{\mathbf{h}^{k-1}_v\} \cup \{\mathbf{h}^{k-1}_u, \forall u \in \mathcal{N}_{\dashv(v)}\}) \tag{6.4a}$$

$$\mathbf{h}^k_{\mathcal{N}_{\vdash(v)}} = \text{MEAN}(\{\mathbf{h}^{k-1}_v\} \cup \{\mathbf{h}^{k-1}_u, \forall u \in \mathcal{N}_{\vdash(v)}\}) \tag{6.4b}$$

For the semantics-aware dynamic graph we compute a weighted average for aggregation where the weights come from the normalized adjacency matrices $\mathbf{A}^{\dashv}$ and $\mathbf{A}^{\vdash}$, defined as,

$$\mathbf{h}^k_{\mathcal{N}_{\dashv(v)}} = \sum_{\forall u \in \mathcal{N}_{\dashv(v)}} \mathbf{a}^{\dashv}_{v,u} \mathbf{h}^{k-1}_u \tag{6.5a}$$

$$\mathbf{h}^k_{\mathcal{N}_{\vdash(v)}} = \sum_{\forall u \in \mathcal{N}_{\vdash(v)}} \mathbf{a}^{\vdash}_{v,u} \mathbf{h}^{k-1}_u \tag{6.5b}$$

While [198] learn separate node embeddings for both directions independently, we opt to fuse information aggregated in two directions at each hop, which we find works better in general.

$$\mathbf{h}^k_{\mathcal{N}_{(v)}} = \text{Fuse}(\mathbf{h}^k_{\mathcal{N}_{\dashv(v)}}, \mathbf{h}^k_{\mathcal{N}_{\vdash(v)}}) \tag{6.6}$$

We design the fusion function as a gated sum of two information sources,

$$\text{Fuse}(\mathbf{a}, \mathbf{b}) = \mathbf{z} \odot \mathbf{a} + (1 - \mathbf{z}) \odot \mathbf{b} \tag{6.7a}$$

$$\mathbf{z} = \sigma(\mathbf{W}_z[\mathbf{a}; \mathbf{b}; \mathbf{a} \odot \mathbf{b}; \mathbf{a} - \mathbf{b}] + \mathbf{b}_z) \tag{6.7b}$$

where $\odot$ is the component-wise multiplication, σ is a sigmoid function, and $\mathbf{z}$ is a gating vector.

Finally, a Gated Recurrent Unit (GRU) [78] is used to update the node embeddings by incorporating the aggregation information.

$$\mathbf{h}_v^k = \text{GRU}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}_{(v)}}^k) \tag{6.8}$$

After n hops of GNN computation, where n is a hyperparameter, we obtain the final state embedding $\mathbf{h}_v^n$ for node v. To compute the graph-level embedding, we first apply a linear projection to the node embeddings, and then apply max-pooling over all node embeddings to get a d-dim vector $\mathbf{h}^{\mathcal{G}}$.

6.2.3.3 *RNN Decoder*

On the decoder side, we adopt the same model architecture as other state-of-the-art Seq2Seq models where an attention-based [79], [140] LSTM decoder with copy [154], [155] and coverage mechanisms [41], [156] is employed. The decoder takes the graph-level embedding $\mathbf{h}^{\mathcal{G}}$ followed by two separate fully-connected layers as initial hidden states (i.e., $\mathbf{c}_0$ and $\mathbf{s}_0$) and the node embeddings $\{\mathbf{h}_v^n, \forall v \in \mathcal{G}\}$ as the attention memory, and generates the output sequence one word at a time.

More specifically, at each decoding step t, an attention mechanism learns to attend to the most relevant words in the input sequence, and computes a context vector $\mathbf{h}_t^*$ based on the current decoding state $\mathbf{s}_t$, the current coverage vector $\mathbf{c}^t$ and the attention memory. In addition, the generation probability $p_{\text{gen}} \in [0, 1]$ is calculated from the context vector $\mathbf{h}_t^*$, the decoder state $\mathbf{s}_t$ and the decoder input y_{t-1}. Next, p_{gen} is used as a soft switch to choose between generating a word from the vocabulary, or copying a word from the input sequence. We dynamically maintain an extended vocabulary which is the union of the usual vocabulary and all words appearing in a batch of source examples (i.e., passages and answers). Finally, in order to encourage the decoder to utilize the diverse components of the input sequence, a

coverage mechanism is applied. At each step, we maintain a coverage vector $\mathbf{c}^t$, which is the sum of attention distributions over all previous decoder time steps. A coverage loss is also computed to penalize repeatedly attending to the same locations of the input sequence.

6.2.4 Hybrid Evaluator

It has been observed that optimizing such cross-entropy based training objectives for sequence learning does not always produce the best results on discrete evaluation metrics [81], [109], [110]. There are two main limitations of this optimizing method: i) exposure bias, meaning that a model has access to the ground-truth sequence up to the next token during training but does not have such supervision when testing, resulting in accumulated errors; ii) evaluation discrepancy between training and testing, meaning that a model is optimized with cross-entropy loss during training while evaluated with discrete evaluation metrics during testing. To tackle these issues, some recent QG approaches [84], [89] directly optimize evaluation metrics using REINFORCE. We further use a mixed objective function with both syntactic and semantic constraints for guiding text generation. In particular, we present a hybrid evaluator with a mixed objective function that combines both cross-entropy loss and RL loss in order to ensure the generation of syntactically and semantically valid text.

For the RL part, we employ the self-critical sequence training (SCST) algorithm [283] to directly optimize the evaluation metrics. SCST is an efficient REINFORCE algorithm that utilizes the output of its own test-time inference algorithm to normalize the rewards it experiences. In SCST, at each training iteration, the model generates two output sequences: the sampled output Y^s, produced by multinomial sampling, that is, each word y_t^s is sampled according to the likelihood $P(y_t|X, y_{<t})$ predicted by the generator, and the baseline output $\hat{Y}$, obtained by greedy search, that is, by maximizing the output probability distribution at each decoding step. We define $r(Y)$ as the reward of an output sequence Y, computed by comparing it to corresponding ground-truth sequence Y^* with some reward metrics. The loss function is defined as:

$$\mathcal{L}_{rl} = (r(\hat{Y}) - r(Y^s)) \sum_t \log P(y_t^s|X, y_{<t}^s) \tag{6.9}$$

As we can see, if the sampled output has a higher reward than the baseline one, we maximize its likelihood, and vice versa.

One of the key factors for RL is to pick the proper reward function. To take syntactic and semantic constraints into account, we consider the following metrics as our reward functions:

Evaluation metric as reward function. We use one of our evaluation metrics, BLEU-4, as our reward function f_{eval}, which lets us directly optimize the model towards the evaluation metrics.

Semantic metric as reward function. One drawback of some evaluation metrics like BLEU is that they do not measure meaning, but only reward systems that have exact n-gram matches in the reference system. To make our reward function more effective and robust, we additionally use word mover's distance (WMD) as a semantic reward function f_{sem}. WMD is the state-of-the-art approach to measure the dissimilarity between two sentences based on word embeddings [289]. Following [290], we take the negative of the WMD distance between a generated sequence and the ground-truth sequence and divide it by the sequence length as its semantic score.

We define the final reward function as $r(Y) = f_{\text{eval}}(Y, Y^*) + \alpha f_{\text{sem}}(Y, Y^*)$ where α is a scalar.

6.2.5 Training and Testing

We train our model in two stages. In the first state, we train the model using regular cross-entropy loss, defined as,

$$\mathcal{L}_{lm} = \sum_t -\log P(y_t^*|X, y_{<t}^*) + \lambda \text{ covloss}_t \tag{6.10}$$

where y_t^* is the word at the t-th position of the ground-truth output sequence and covloss_t is the coverage loss defined as $\sum_i min(a_i^t, c_i^t)$, with a_i^t being the i-th element of the attention vector over the input sequence at time step t. Scheduled teacher forcing [282] is adopted to alleviate the exposure bias problem. In the second stage, we fine-tune the model by optimizing a mixed objective function combining both cross-entropy loss and RL loss, defined as,

$$\mathcal{L} = \gamma \mathcal{L}_{rl} + (1 - \gamma)\mathcal{L}_{lm} \tag{6.11}$$

where γ is a scaling factor controling the trade-off between cross-entropy loss and RL loss. During the testing phase, we use beam search to generate final predictions.

6.3 Experiments

We evaluate our proposed model against state-of-the-art methods on the SQuAD dataset [149]. Our full models have two variants G2S$_{sta}$+BERT+RL and G2S$_{dyn}$+BERT+RL which adopts static graph construction or dynamic graph construction, respectively. The implementation of our model is publicly available at `https://github.com/hugochan/RL-based-Graph2Seq-for-NQG`.

6.3.1 Baseline Methods

We compare our models against the following baseline methods in our experiments: i) Transformer [108], ii) SeqCopyNet [291], iii) NQG++ [74], iv) MPQG+R [89], v) AFPQA [91], vi) ASs2s [90], vii) s2sa-at-mp-gsa [87], and viii) CGC-QG [88]. Experiments on baselines followed by * are conducted using released code. Results of other baselines are taken from the corresponding papers, with unreported metrics marked as –. Detailed descriptions of the baselines are provided next.

Transformer [108] We included a Transformer-based Seq2Seq model augmented with attention and copy mechanisms. We used the open source implementation provided by the OpenNMT [284] library and trained the model from scratch. Surprisingly, this baseline performed very poorly on the benchmarks (as we will see later) even though we conducted moderate hyperparameter search and trained the model for a large amount of epochs. We suspect this might be partially because this method is very sensitive to hyperparameters as reported by [284] and probably data-hungry on this task. We conjecture that better performance might be expected by extensively searching the hyperparameters and using a pretrained transformer model.

SeqCopyNet [291] proposed an extension to the copy mechanism which learns to copy not only single words but also sequences from the input sentence.

NQG++ [74] proposed an attention-based Seq2Seq model equipped with a copy mechanism and a feature-rich encoder to encode answer position, POS and NER tag information.

MPQG+R [89] proposed an RL-based Seq2Seq model equipped with a multi-perspective matching encoder to incorporate answer information. Copy and coverage mechanisms are applied.

`https://opennmt.net/OpenNMT-py/FAQ.html`

AFPQA [91] consists of an answer-focused component which generates an interrogative word matching the answer type, and a position-aware component which is aware of the position of the context words when generating a question by modeling the relative distance between the context words and the answer.

ASs2s [90] proposed an answer-separated Seq2Seq model which treats the passage and the answer separately.

s2sa-at-mp-gsa [87] proposed a model which contains a gated attention encoder and a maxout pointer decoder to tackle the challenges of processing long input sequences. For fair comparison, we report the results of the sentence-level version of their model to match with our settings.

CGC-QG [88] proposed a multi-task learning framework to guide the model to learn the accurate boundaries between copying and generation.

6.3.2 Data and Metrics

SQuAD contains more than 100K questions posed by crowd workers on 536 Wikipedia articles. Since the test set of the original SQuAD is not publicly available, the accessible parts (≈90%) are used as the entire dataset in our experiments. For fair comparison with previous methods, we evaluated our model on both data split-1 [75] that contains 75,500/17,934/11,805 (train/development/test) examples and data split-2 [74] that contains 86,635/8,965/8,964 examples.

Following previous works, we use BLEU-4 [113], METEOR [114], ROUGE-L [115] and Q-BLEU1 [118] as our evaluation metrics. Initially, BLEU-4 and METEOR were designed for evaluating machine translation systems and ROUGE-L was designed for evaluating text summarization systems. Recently, Q-BLEU1 was designed for better evaluating question generation systems, which was shown to correlate significantly better with human judgments compared to existing metrics.

Besides automatic evaluation, we also conducted a small-scale (i.e., 50 random examples per system) human evaluation on the split-2 data. We asked 5 human evaluators to give feedback on the quality of questions generated by a set of anonymized competing systems. In each example, given a triple containing a source passage, a target answer and an

https://www.cs.rochester.edu/~lsong10/downloads/nqg_data.tgz
https://res.qyzhou.me/redistribute.zip

anonymised system output, they were asked to rate the quality of the output by answering the following three questions: i) is this generated question syntactically correct? ii) is this generated question semantically correct? and iii) is this generated question relevant to the passage? For each evaluation question, the rating scale is from 1 to 5 where a higher score means better quality (i.e., 1: Poor, 2: Marginal, 3: Acceptable, 4: Good, 5: Excellent). Responses from all evaluators were collected and averaged.

6.3.3 Model Settings

We keep and fix the 300-dim GloVe vectors for the most frequent 70,000 words in the training set. We compute the 1024-dim BERT embeddings on the fly for each word in text using a (trainable) weighted sum of all BERT layer outputs. The embedding sizes of case, POS and NER tags are set to 3, 12 and 8, respectively. We set the hidden state size of BiLSTM to 150 so that the concatenated state size for both directions is 300. The size of all other hidden layers is set to 300. We apply a variational dropout [288] rate of 0.4 after word embedding layers and 0.3 after RNN layers. We set the neighborhood size to 10 for dynamic graph construction. The number of GNN hops is set to 3. During training, in each epoch, we set the initial teacher forcing probability to 0.75 and exponentially increase it to $0.75 * 0.9999^i$ where i is the training step. In addition, partial teacher forcing is adopted, which means that when generating a sequence, some steps can be teacher forced and some not. We set α in the reward function to 0.1, γ in the mixed loss function to 0.99, and the coverage loss ratio λ to 0.4. The beam search width is set to 5. During beam search, we set the minimal and maximal output sequence lengths as 4 and 40, respectively. The maximal number of decoding steps is set to 41. We use Adam [274] as the optimizer, and the learning rate is set to 0.001 in the pretraining stage and 0.00001 in the fine-tuning stage. We reduce the learning rate by a factor of 0.5 if the validation BLEU-4 score stops improving for three epochs. We stop the training when no improvement is seen for 10 epochs. We clip the gradient at length 10. The batch size is set to 60 and 50 on data split-1 and split-2, respectively. All hyperparameters are tuned on the development set.

6.3.4 Experimental Results

Table 6.1 shows the automatic evaluation results comparing our proposed models against other state-of-the-art baseline methods. First of all, we can see that both of our full

Table 6.1: Automatic evaluation results on the SQuAD test set.

Methods	Split-1				Split-2			
	BLEU-4	METEOR	ROUGE-L	Q-BLEU1	BLEU-4	METEOR	ROUGE-L	Q-BLEU1
Transformer	2.56	8.98	26.01	16.70	3.09	9.68	28.86	20.10
SeqCopyNet	–	–	–	–	13.02	–	44.00	–
NQG++	–	–	–	–	13.29	–	–	–
MPQG+R*	14.39	18.99	42.46	52.00	14.71	18.93	42.60	50.30
AFPQA	–	–	–	–	15.64	–	–	–
s2sa-at-mp-gsa	15.32	19.29	43.91	–	15.82	19.67	44.24	–
ASs2s	16.20	19.92	43.96	–	16.17	–	–	–
CGC-QG	–	–	–	–	17.55	21.24	44.53	–
G2S$_{dyn}$+BERT+RL	17.55	21.42	45.59	55.40	18.06	21.53	45.91	55.00
G2S$_{sta}$+BERT+RL	**17.94**	**21.76**	**46.02**	**55.60**	**18.30**	**21.70**	**45.98**	**55.20**

models G2S$_{sta}$+BERT+RL and G2S$_{dyn}$+BERT+RL achieve the new state-of-the-art scores on both data splits and consistently outperform previous methods by a significant margin. This highlights that our RL-based Graph2Seq model, together with the deep alignment network, successfully addresses the three issues we highlighted in Section 6.1. Notably, some previous state-of-the-art methods relied on many heuristic rules and ad-hoc strategies. For instance, CGC-QG [88] annotated clue words in the passage based on word frequency and overlapping, masked out low-frequency passage word embeddings, and reduced the target output vocabulary to boost the model performance. ASs2s [90] replaced the target answer in the original passage with a special token. However, our proposed model does not rely on any hand-crafted rules or ad-hoc strategies, and is fully end-to-end trainable. Between these two variants, G2S$_{sta}$+BERT+RL outperforms G2S$_{dyn}$+BERT+RL on all the metrics.

As shown in Table 6.2, we conducted a human evaluation study to assess the quality of the questions generated by our model, the baseline method MPQG+R, and the ground-truth data in terms of syntax, semantics and relevance metrics. We can see that our best performing model achieves good results even compared to the ground-truth, and outperforms the strong baseline method MPQG+R. Our error analysis shows that main syntactic error occurs in repeated/unknown words in generated questions. Further, the slightly lower quality on semantics also impacts the relevance.

Table 6.2: Human evaluation results (± standard deviation) on the SQuAD split-2 test set.

Methods	Syntactically correct	Semantically correct	Relevant
MPQG+R*	4.34 (0.15)	4.01 (0.23)	3.21 (0.31)
G2S$_{sta}$+BERT+RL	4.41 (0.09)	4.31 (0.12)	3.79 (0.45)
Ground-truth	**4.74 (0.14)**	**4.74 (0.19)**	**4.25 (0.38)**

6.3.5 Ablation Study

As shown in Table 6.3, we perform an ablation study to systematically assess the impact of different model components (e.g., BERT, RL, DAN, and BiGGNN) for two proposed full model variants (static vs dynamic) on the SQuAD split-2 test set. It confirms our finding that syntax-based static graph construction (G2S$_{sta}$+BERT+RL) performs better than semantics-aware dynamic graph construction (G2S$_{dyn}$+BERT+RL) in almost every setting. However, it may be too early to conclude which one is the method of choice for QG. On the one hand, an advantage of static graph construction is that useful domain knowledge can be hard-coded into the graph, which can greatly benefit the downstream task. However, it might suffer if there is a lack of prior knowledge for a specific domain knowledge. On the other hand, dynamic graph construction does not need any prior knowledge about the hidden structure of text, and only relies on the attention matrix to capture these structured information, which provides an easy way to achieve a decent performance. One interesting direction is to explore effective ways of combining both static and dynamic graphs.

By turning off the Deep Alignment Network (DAN), the BLEU-4 score of G2S$_{sta}$ (similarly for G2S$_{dyn}$) dramatically drops from 16.96% to 12.62%, which indicates the importance of answer information for QG and shows the effectiveness of DAN. This can also be verified by comparing the performance between the DAN-enhanced Seq2Seq model (16.14 BLEU-4 score) and other carefully designed answer-aware Seq2Seq baselines such as NQG++ (13.29 BLEU-4 score), MPQG+R (14.71 BLEU-4 score) and AFPQA (15.82 BLEU-4 score). Further experiments demonstrate that both word-level (G2S$_{sta}$ w/ DAN-word only) and contextual-level (G2S$_{sta}$ w/ DAN-contextual only) answer alignments in DAN are helpful.

We can see the advantages of Graph2Seq learning over Seq2Seq learning on this task by comparing the performance between G2S$_{sta}$ and Seq2Seq. Compared to Seq2Seq based QG methods that completely ignore hidden structure information in the passage, our Graph2Seq based method is aware of more hidden structure information such as semantic similarity

Table 6.3: Ablation study on the SQuAD split-2 test set.

Methods	BLEU-4	Methods	BLEU-4
G2S$_{dyn}$+BERT+RL	18.06	G2S$_{dyn}$ w/o feat	16.51
G2S$_{sta}$+BERT+RL	**18.30**	G2S$_{sta}$ w/o feat	16.65
G2S$_{sta}$+BERT-fixed+RL	18.20	G2S$_{dyn}$ w/o DAN	12.58
G2S$_{dyn}$+BERT	17.56	G2S$_{sta}$ w/o DAN	12.62
G2S$_{sta}$+BERT	18.02	G2S$_{sta}$ w/ DAN-word only	15.92
G2S$_{sta}$+BERT-fixed	17.86	G2S$_{sta}$ w/ DAN-contextual only	16.07
G2S$_{dyn}$+RL	17.18	G2S$_{sta}$ w/ GGNN-forward	16.53
G2S$_{sta}$+RL	17.49	G2S$_{sta}$ w/ GGNN-backward	16.75
G2S$_{dyn}$	16.81	G2S$_{sta}$ w/o BiGGNN, w/ Seq2Seq	16.14
G2S$_{sta}$	16.96	G2S$_{sta}$ w/o BiGGNN, w/ GCN	14.47

between any pair of words that are not directly connected or syntactic relationships between two words captured in a dependency parsing tree. In our experiments, we also observe that doing both forward and backward message passing in the GNN encoder is beneficial. Surprisingly, using GCN [193] as the graph encoder (and converting the input graph to an undirected graph) does not provide good performance. In addition, fine-tuning the model using REINFORCE can further improve the model performance in all settings (i.e., w/ and w/o BERT), which shows the benefits of directly optimizing the evaluation metrics. Besides, we find that the pretrained BERT embedding has a considerable impact on the performance and fine-tuning BERT embedding even further improves the performance, which demonstrates the power of large-scale pretrained language models. Incorporating common linguistic features (i.e., case, POS, NER) also helps the overall performance to some extent.

6.3.6 Case Study

In Table 6.4, we further show a few examples that illustrate the quality of generated text given a passage under different ablated systems. As we can see, incorporating answer information helps the model identify the answer type of the question to be generated, and thus makes the generated questions more relevant and specific. Also, we find our Graph2Seq model can generate more complete and valid questions compared to the Seq2Seq baseline. We think it is because a Graph2Seq model is able to exploit the rich text structure information better than a Seq2Seq model. Lastly, it shows that fine-tuning the model using REINFORCE can improve the quality of the generated questions.

Table 6.4: Generated questions on SQuAD split-2 test set. Target answers are underlined.

Passage: for the successful execution of a project , effective planning is essential .
Gold: what is essential for the successful execution of a project ?
G2S$_{sta}$ w/o BiGGNN (Seq2Seq): what type of planning is essential for the project ?
G2S$_{sta}$ w/o DAN: what type of planning is essential for the successful execution of a project ?
G2S$_{sta}$: what is essential for the successful execution of a project ?
G2S$_{sta}$+BERT: what is essential for the successful execution of a project ?
G2S$_{sta}$+BERT+RL: what is essential for the successful execution of a project ?
G2S$_{dyn}$+BERT+RL: what is essential for the successful execution of a project ?
Passage: the church operates three hundred sixty schools and institutions overseas .
Gold: how many schools and institutions does the church operate overseas ?
G2S$_{sta}$ w/o BiGGNN (Seq2Seq): how many schools does the church have ?
G2S$_{sta}$ w/o DAN: how many schools does the church have ?
G2S$_{sta}$: how many schools and institutions does the church have ?
G2S$_{sta}$+BERT: how many schools and institutions does the church have ?
G2S$_{sta}$+BERT+RL: how many schools and institutions does the church operate ?
G2S$_{dyn}$+BERT+RL: how many schools does the church operate ?

6.3.7 Sensitivity Analysis of Hyperparameters

To study the effect of the number of GNN hops, we conduct experiments on the G2S$_{sta}$ model on the SQuAD split-2 data. Fig. 6.3 shows that our model is not very sensitive to the number of GNN hops and can achieve reasonably good results with various number of hops.

6.4 Conclusion and Future Work

We proposed a novel RL based Graph2Seq model for QG, where the answer information is utilized by an effective Deep Alignment Network and a novel bidirectional GNN is proposed to process the directed passage graph. Our two-stage training strategy benefits from both cross-entropy based and REINFORCE based sequence training. We also explore both static and dynamic graph construction from text, and systematically investigate and analyze the performance difference between the two. On the SQuAD dataset, our method outperforms existing methods by a significant margin and achieves the new state-of-the-art results.

Future directions include investigating more effective ways of automatically learning graph structures from text and exploiting Graph2Seq models for question generation from structured data like knowledge graphs or tables.

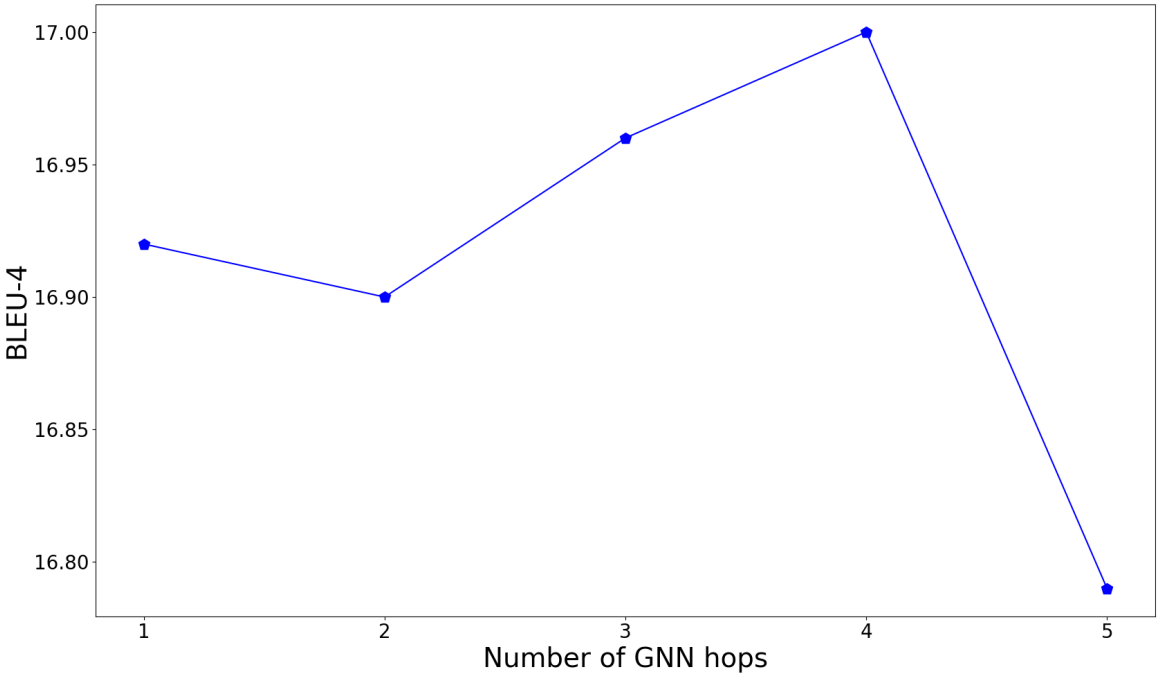

Fig. 6.3: Effect of the number of GNN hops.

CHAPTER 7
CONCLUSION AND FUTURE WORK

7.1 Conclusion

In this dissertation, we presented our work on question answering over different knowledge sources including a knowledge base and free-form text. We identify and address three main challenges of question answering, which include the lexical gap between natural language questions and the underlying knowledge sources, complex reasoning and conversational question answering. In addition, we presented our work on question generation which is a dual task of question answering, and can be leveraged to help the QA tasks by providing more training data. We identity and address three main challenges of question generation, which include effect context modeling, target answer utilization and sequence learning model training. Here, we briefly summarize our work and discuss the limitations.

In Chapter 3, we proposed a novel and effective information retrieval based approach for the task of question answering over knowledge bases. The proposed BAMnet model is able to perform multi-hop reasoning in a knowledge base and requires no external resources and very few hand-crafted features. Experiments on a popular KBQA benchmark show that our model significantly outperforms previous information retrieval based methods while remaining competitive with handcrafted semantic parsing based methods. However, the proposed model have several limitations. Although it can perform multi-hop reasoning in a knowledge base, it cannot handle complex constraints (e.g., ordinal and aggregation constraints) and symbolic operations (e.g., comparison and quantitative operations). Besides, it can only answer single-turn questions and is not aware of conversation context.

In Chapter 4, in order to capture conversation history more effectively in the task of conversational machine reading comprehension, we proposed a novel graph neural network based method which is able to model conversational flow in a dialog. On three widely-used benchmarks, the proposed model shows superior performance compared to existing state-of-the-art methods. Our interpretability analysis shows that our model can offer good interpretability for the reasoning process. However, one limitation of the proposed model is its incapability of performing complex reasoning since it is just doing sophisticated semantic matching at its best. Moreover, in Chapter 4, we define conversation flow as a sequence of latent states in the dialog. However, we did not learn explicit representations of the

conversation flow in a dialog, instead, we modeled it as changes of passage representations between consecutive conversation turns. It is not clear what is the best way to represent or model conversation flow.

In Chapter 5, in order to better capture the structure information of KG subgraphs, we applied a bidirectional Graph2Seq model to encode the KG subgraph. Furthermore, we proposed to enhance the RNN decoder with the novel node-level copying mechanism to allow directly copying node attributes from the KG subgraph to the output question. Both automatic and human evaluation results demonstrate that our model achieves new state-of-the-art scores, outperforming existing methods by a significant margin on two QG benchmarks. Experiments also show that our QG model can consistently benefit the QA task as a mean of data augmentation. However, how to effectively utilize the target answer information remains a open question. Our experiments show that a simple trick of incorporating the answer markup can help improve the model performance. It will be interesting to see if we can further improve the performance by leveraging more advanced techniques of utilizing answer information.

In Chapter 6, we proposed a reinforcement learning (RL) based Graph2Seq model for QG from text. Our model consists of a Graph2Seq generator with a novel Bidirectional Gated Graph Neural Network based encoder to embed the passage, and a hybrid evaluator with a mixed objective combining both the cross-entropy loss and RL loss to ensure the generation of syntactically and semantically valid text. We also introduce an effective Deep Alignment Network for incorporating the answer information into the passage at both the word and contextual levels. Our model is end-to-end trainable and achieves new state-of-the-art scores, outperforming existing methods by a significant margin on the standard SQuAD benchmark. Our experiments show that modeling the rich structure information of the passage can improve the model performance. However, it is not clear which kind of graph representation is best for the QG task. It will be interesting to conduct a thorough comparative study on the performance impact of different kinds of static and dynamic graph representation techniques on the QG task.

7.2 Future Work

Next, we discuss several future directions for our current work.

7.2.1 Question Answering

7.2.1.1 *Complex Question Answering*

As aforementioned, our proposed methods are limited in terms of the capability of performing complex reasoning. Indeed, complex question answering is a very challenging yet rewarding problem since making machines perform complex reasoning is a big step towards human-level artificial intelligence.

Complex question answering as program induction. We believe that treating complex question answering as a program induction problem is a generic and promising direction. Considering the limitation of neural networks that everything must be embeded into a vector and that they are incapable of performing precise program execution, combining neural networks and symbolic methods is promising. While neural networks are not good at program execution, we believe they are still effective ways of program search. We believe there are two important components in this paradigm. One is to explore more efficient and effective ways of sampling symbolic operations in the program space probably by introducing some pruning strategies or inductive biases. The other is to find more efficient and effective ways of training the models, e.g., better reinforcement learning algorithms or pretraining strategies to boost reinforcement learning.

Complex question answering as graph reasoning. We would also like to explore the possibilities of applying graph neural networks for complex reasoning. Graph neural networks provide us a systematic way to introduce strong inductive biases into models, which might help efficient search in the program space. One idea is to dynamically construct a question-aware graph which consists of symbolic operations as nodes. Therefore, searching in the program space is equivalent to doing random walk in the graph, which will largely reduce the search space.

7.2.1.2 *Conversational Question Answering*

We believe that modeling conversation flow is important for conversational question answering, and more broadly dialog systems. However, as we mentioned earlier, it is not clear what is the best way to represent or model conversation flow. Another challenge here is that we do not have very good ways to evaluate whether a model successfully captures the conversation flow or not. The interpretability analysis we did in Chapter 4 is a good attempt but is still not satisfying since it is implicit. In the future, we would like to explore more

effective ways of modeling conversation flow in the dialog as well as better ways to visualize them.

7.2.1.3 *Question Answering from Multimodal Data*

Most previous works on QA focused on question answering from a single knowledge source such as text or images. Recently, multimodal question answering has gained increasing attention as it provides human users a more natural and effective way for information seeking. [292] released a multimodal question answering dataset in the cultural heritage domain that contains multimodal content about the fascinating old-Egyptian Amarna period, including images of typical artworks, documents about these artworks (containing images) and over 800 multimodal queries integrating visual and textual questions. [293] introduced the task of Multi-Modal Machine Comprehension, which aims at answering multimodal questions given a context of text, diagrams and images. [294] considered the task of multimodal question answering over structured data, in which a user supplies not just a natural language query but also an image.

7.2.2 Question Generation

7.2.2.1 *Personalized Question Generation*

Almost all existing QG systems are non-personalized, and are not able to adapt their generated questions to different users. We believe that personalized question generation can be beneficial for educational assessment and dialog systems that require certain amount of personalization. In order to build a personalized QG system, one might need to take the user profile as additional input to the QG system, and integrate it with an additional user profile modeling component.

7.2.2.2 *Conversational Question Generation*

Most existing QG systems can only generate a single question from the input context. It will be interesting to see QG systems that can actually generate a sequence of related questions from some long/large context. And this can be beneficial for helping conversational question answering systems and dialog systems. Recent studies have explored the task of conversational question generation. [295] focused on generating a question based on a passage and a conversation history (i.e., previous turns of question-answer pairs). To this end, they

introduced an encoder-decoder framework which incorporates a reasoning procedure in a dynamic manner to better understand what has been asked and what to ask next about the passage. [296] proposed to first locate the question focus in the passage, and then identify the question pattern that leads the sequential generation of the words in a question.

7.2.2.3 *Question Generation from Multimodal Data*

Similar to the situation in QA, almost all previous works on QG focused on question generation from single knowledge source. A natural follow-up question is "can we do question generation from multimodal data?". Although to the best of our knowledge, this topic has never been studied yet, we think it is worth exploring. Just as the classic single-source QG, we believe QG from multimodal data will have useful applications in helping multimodal question answering, generating comprehensive practice exercises and assessments for educational purposes, and helping dialog systems to kick-start and continue a conversation with human users. Moreover, by utilizing rich context information, we might expect a QG system to generate more complex and realistic questions.

7.2.2.4 *Joint Learning of QA & QG*

QG is a dual task of QA. Previously studies have shown that QG can help QA by providing more training data [86], [94], [101]–[103], and QA can help QG by providing rich reward signals on "answerability" [101]. A natural follow-up question is "can we jointly train better QA and QG systems that can help each other?". There is a line of research efforts on jointly training QA and QG systems [92], [93], [297]–[302], but this requires further work.

BIBLIOGRAPHY

[1] W. A. Woods, "Progress in natural language understanding: an application to lunar geology," in *Amer. Federation of Inf. Process. Societies: 1973 Nat. Comput. Conf.*, Jun. 4-8, 1973, pp. 441–450.

[2] J. Kupiec, "Murax: A robust linguistic approach for question answering using an on-line encyclopedia," in *Proc. 16th Annu. Int. Conf. Res. and Develop. Inf. Retrieval*, Jun. 27 - Jul. 1, 1993, pp. 181–190.

[3] J. Berant, A. Chou, R. Frostig, and P. Liang, "Semantic parsing on freebase from question-answer pairs," in *Proc. 2013 Conf. Empirical Meth. Natural Lang. Process.*, Oct. 18-21, 2013, pp. 1533–1544.

[4] A. Bordes, J. Weston, and N. Usunier, "Open question answering with weakly supervised embedding models," in *Mach. Learn. and Knowl. Discovery in Databases - Eur. Conf.*, Sep. 15-19, 2014, pp. 165–180.

[5] A. Bordes, S. Chopra, and J. Weston, "Question answering with subgraph embeddings," in *Proc. 2014 Conf. Empirical Meth. Natural Lang. Process*, Oct. 25-29, 2014, pp. 615—-620.

[6] P. Yin, Z. Lu, H. Li, and B. Kao, "Neural enquirer: Learning to query tables with natural language," 2015, *arXiv:1512.00965*.

[7] S. Reed and N. De Freitas, "Neural programmer-interpreters," 2015, *arXiv:1511.06279*.

[8] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi, "Bidirectional attention flow for machine comprehension," 2016, *arXiv:1611.01603*.

[9] L. Dong, F. Wei, M. Zhou, and K. Xu, "Question answering over freebase with multi-column convolutional neural networks," in *Proc. 53rd Annu. Meeting Assoc. Comput. Ling. and 7th Int. Joint Conf. Natural Lang. Process.*, vol. 1, Jul. 26-31, 2015, pp. 260–269.

[10] S. Jain, "Question answering over knowledge base using factual memory networks," in *Proc. 2016 Conf. N. Amer. Chap. Assoc. Comput. Ling.: Human Lang. Technol.*, Jun. 12-17, 2016, pp. 109–115.

[11] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "Dbpedia: A nucleus for a web of open data," in *Proc. 6th Int. Semantic Web Conf. and 2nd Asian Semantic Web Conf.*, Nov. 11-15, 2007, vol. 4825, pp. 722–735.

[12] Google, "Freebase data dumps," https://developers.google.com/freebase (Accessed May 2, 2018).

[13] F. Mahdisoltani, J. Biega, and F. M. Suchanek, "Yago3: A knowledge base from multilingual wikipedias," in *Proc. 7th Biennial Conf. Innovative Data Syst. Res.*, Jan. 4-7, 2015.

[14] D. Vrandečić and M. Krötzsch, "Wikidata: a free collaborative knowledgebase," *Commun. of the ACM*, vol. 57, no. 10, pp. 78–85, Sep. 2014.

[15] M.-C. Yang, N. Duan, M. Zhou, and H.-C. Rim, "Joint relational embeddings for knowledge-based question answering," in *Proc. 2014 Conf. Empirical Meth. Natural Lang. Process.*, Oct. 25-29, 2014, pp. 645–650.

[16] A. Bordes, N. Usunier, S. Chopra, and J. Weston, "Large-scale simple question answering with memory networks," 2015, *arXiv:1506.02075*.

[17] K. Xu, S. Reddy, Y. Feng, S. Huang, and D. Zhao, "Question answering on freebase via relation extraction and textual evidence," 2016, *arXiv:1603.00957*.

[18] Y. Hao, Y. Zhang, K. Liu, S. He, Z. Liu, H. Wu, and J. Zhao, "An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge," in *Proc. 55th Annu. Meeting Assoc. Comput. Ling.*, vol. 1, Jul. 30 - Aug. 4, 2017, pp. 221–231.

[19] A. Saha, V. Pahuja, M. M. Khapra, K. Sankaranarayanan, and S. Chandar, "Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph," in *Proc. 32nd AAAI Conf. Artif. Intell.*, Feb. 2-7, 2018, pp. 705–713.

[20] P. Trivedi, G. Maheshwari, M. Dubey, and J. Lehmann, "Lc-quad: A corpus for complex question answering over knowledge graphs," in *Proc. 16th Int. Semantic Web Conf.*, vol. 10588, Oct. 21-25, 2017, pp. 210–218.

[21] Y. Su, H. Sun, B. Sadler, M. Srivatsa, I. Gür, Z. Yan, and X. Yan, "On generating characteristic-rich question sets for QA eval." in *Proc. 2016 Conf. Empirical Meth. Natural Lang. Process.*, Nov. 1-4, 2016, pp. 562–572.

[22] Y. Zhang, H. Dai, Z. Kozareva, A. J. Smola, and L. Song, "Variational reasoning for question answering with knowledge graph," in *Proc. 32nd AAAI Conf. Artif. Intell.*, Feb. 2-7, 2018, pp. 6069–6076.

[23] D. Dua, Y. Wang, P. Dasigi, G. Stanovsky, S. Singh, and M. Gardner, "Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs," 2019, *arXiv:1903.00161*.

[24] P. Pasupat and P. Liang, "Compositional semantic parsing on semi-structured tables," 2015, *arXiv:1508.00305*.

[25] V. Zhong, C. Xiong, and R. Socher, "Seq2sql: Generating structured queries from natural language using reinforcement learning," 2017, *arXiv:1709.00103*.

[26] A. Neelakantan, Q. V. Le, and I. Sutskever, "Neural programmer: Inducing latent programs with gradient descent," 2015, *arXiv:1511.04834*.

[27] A. Neelakantan, Q. V. Le, M. Abadi, A. McCallum, and D. Amodei, "Learning a natural language interface with neural programmer," 2016, *arXiv:1611.08945*.

[28] C. Liang, J. Berant, Q. Le, K. D. Forbus, and N. Lao, "Neural symbolic machines: Learning semantic parsers on freebase with weak supervision," in *Proc. 55th Annu. Meeting Assoc. Comput. Ling.*, vol. 1, Jul. 30 - Aug. 4, 2017, pp. 23–33.

[29] J. Krishnamurthy, P. Dasigi, and M. Gardner, "Neural semantic parsing with type constraints for semi-structured tables," in *Proc. 2017 Conf. Empirical Meth. Natural Lang. Process.*, Sep. 9-11, 2017, pp. 1516–1526.

[30] F. Yang, Z. Yang, and W. W. Cohen, "Differentiable learning of logical rules for knowledge base reasoning," in *Advances Neural Inf. Process. Syst.*, Dec. 4-9, 2017, pp. 2319–2328.

[31] L. Mou, Z. Lu, H. Li, and Z. Jin, "Coupling distributed and symbolic execution for natural language queries," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, Aug. 6-11, 2017, pp. 2518–2526.

[32] S. Li, H. Xu, and Z. Lu, "Generalize symbolic knowledge with neural rule engine," 2018, *arXiv:1808.10326*.

[33] M. Iyyer, W.-t. Yih, and M.-W. Chang, "Search-based neural structured learning for sequential question answering," in *Proc. 55th Annu. Meeting Assoc. Comput. Ling.*, vol. 1, Jul. 30 - Aug. 4, 2017, pp. 1821–1831.

[34] X. Li, S. Panda, J. Liu, and J. Gao, "Microsoft dialogue challenge: Building end-to-end task-completion dialogue systems," 2018, *arXiv:1807.11125*.

[35] C. Gunasekara, J. K. Kummerfeld, L. Polymenakos, and W. Lasecki, "Dstc7 task 1: Noetic end-to-end response selection," in *Proc. 1st Workshop on NLP for Conversational AI*, 2019, pp. 60–67.

[36] L. E. Asri, H. Schulz, S. Sharma, J. Zumer, J. Harris, E. Fine, R. Mehrotra, and K. Suleman, "Frames: A corpus for adding memory to goal-oriented dialogue systems," 2017, *arXiv:1704.00057*.

[37] S. Reddy, D. Chen, and C. D. Manning, "Coqa: A conversational question answering challenge," 2018, *arXiv:1808.07042*.

[38] E. Choi, H. He, M. Iyyer, M. Yatskar, W.-t. Yih, Y. Choi, P. Liang, and L. Zettlemoyer, "Quac: Question answering in context," 2018, *arXiv:1808.07036*.

[39] M. Saeidi, M. Bartolo, P. Lewis, S. Singh, T. Rocktäschel, M. Sheldon, G. Bouchard, and S. Riedel, "Interpretation of natural language rules in conversational machine reading," 2018, *arXiv:1809.01494*.

[40] J. A. Campos, A. Otegi, A. Soroa, J. Deriu, M. Cieliebak, and E. Agirre, "Conversational qa for faqs," in *3rd Conversational AI Workshop at the Conf. Neural Inf. Process. Syst.*, Dec. 8-14, 2019.

[41] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," 2017, *arXiv:1704.04368*.

[42] D. Chen, A. Fisch, J. Weston, and A. Bordes, "Reading wikipedia to answer open-domain questions," 2017, *arXiv:1704.00051*.

[43] M. Yatskar, "A qualitative comparison of coqa, squad 2.0 and quac," 2018, *arXiv:1809.10735*.

[44] J. C. Brown, G. A. Frishkoff, and M. Eskenazi, "Automatic question generation for vocabulary assessment," in *Proc. Conf. Human Lang. Technol. and Empirical Meth. Natural Lang. Process.*, Oct. 6-8, 2005, pp. 819–826.

[45] V. Rus, Z. Cai, and A. C. Graesser, "Experiments on generating questions about facts," in *Int. Conf. Intell. Text Process. and Comput. Ling.*, Feb. 18-24, 2007, pp. 444–455.

[46] V. Rus, Z. Cai, and A. Graesser, "Question generation: Example of a multi-year evaluation campaign," in *Proc. Workshop Question Gener. Shared Task & Eval. Challenge*, 2008.

[47] V. Rus and J. Lester, "The 2nd workshop on question generation," in *Proc. 2009 Conf. Artif. Intell. Educ.*, Jul. 6-10, 2009, pp. 808–808.

[48] V. Rus, B. Wyse, P. Piwek, M. Lintean, S. Stoyanchev, and C. Moldovan, "The first question generation shared task evaluation challenge," in *Proc. 6th Int. Natural Lang. Gener. Conf.*, Jul. 7-9, 2010.

[49] M. Liu, R. A. Calvo, and V. Rus, "Automatic question generation for literature review writing support," in *Int. Conf. Intell. Tutoring Syst.*, Jun. 14-18, 2010, pp. 45–54.

[50] V. Rus, B. Wyse, P. Piwek, M. Lintean, S. Stoyanchev, and C. Moldovan, "Question generation shared task and evaluation challenge–status report," in *Proc. 13th Eur. Workshop Natural Lang. Gener.*, Sep. 28-30, 2011, pp. 318–320.

[51] M. Liu, R. A. Calvo, and V. Rus, "G-asks: An intelligent automatic question generation system for academic writing support," *Dialogue & Discourse*, vol. 3, no. 2, pp. 101–124, Mar. 2012.

[52] V. Rus, B. Wyse, P. Piwek, M. Lintean, S. Stoyanchev, and C. Moldovan, "A detailed account of the first question generation shared task evaluation challenge," *Dialogue & Discourse*, vol. 3, no. 2, pp. 177–204, Mar. 2012.

[53] I. M. Mora and S. P. de la Puente, "Towards automatic generation of question answer pairs from images," in *Vis. Question Answering Challenge Workshop, Conf. Comput. Vision and Pattern Recognit.*, Jun. 2016.

[54] N. Mostafazadeh, I. Misra, J. Devlin, M. Mitchell, X. He, and L. Vanderwende, "Generating natural questions about an image," 2016, *arXiv:1603.06059*.

[55] S. Zhang, L. Qu, S. You, Z. Yang, and J. Zhang, "Automatic generation of grounded visual questions," 2016, *arXiv:1612.06530*.

[56] U. Jain, Z. Zhang, and A. G. Schwing, "Creativity: Generating diverse questions using variational autoencoders," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognit.*, Jul. 21-26, 2017, pp. 6485–6494.

[57] Y. Li, N. Duan, B. Zhou, X. Chu, W. Ouyang, X. Wang, and M. Zhou, "Visual question generation as dual task of visual question answering," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognit.*, Jun. 18-22, 2018, pp. 6116–6124.

[58] Z. Fan, Z. Wei, S. Wang, Y. Liu, and X. Huang, "A reinforcement learning framework for natural question generation using bi-discriminators," in *Proc. 27th Int. Conf. Comput. Ling.*, Aug. 20-26, 2018, pp. 1763–1774.

[59] R. Lebret, D. Grangier, and M. Auli, "Neural text generation from structured data with application to the biography domain," 2016, *arXiv:1603.07771*.

[60] T. Liu, K. Wang, L. Sha, B. Chang, and Z. Sui, "Table-to-text generation by structure-aware seq2seq learning," in *Proc. 32nd AAAI Conf. Artif. Intell.*, Feb. 2-7, 2018, pp. 4881–4888.

[61] J. Bao, D. Tang, N. Duan, Z. Yan, Y. Lv, M. Zhou, and T. Zhao, "Table-to-text: Describing table region with natural language," in *Proc. 32nd AAAI Conf. Artif. Intell.*, Feb. 2-7, 2018, pp. 5020–5027.

[62] T. Liu, F. Luo, Q. Xia, S. Ma, B. Chang, and Z. Sui, "Hierarchical encoder with auxiliary supervision for neural table-to-text generation: Learning better representation for tables," in *Proc. 33rd AAAI Conf. Artif. Intell.*, vol. 3, Jan. 27 - Feb. 1, 2019, pp. 6786–6793.

[63] D. Seyler, M. Yahya, and K. Berberich, "Generating quiz questions from knowledge graphs," in *Proc. 24th Int. Conf. World Wide Web*, May 18-22, 2015, pp. 113–114.

[64] L. Song and L. Zhao, "Question generation from a knowledge base with web exploration," 2016, *arXiv:1610.03807*.

[65] D. Seyler, M. Yahya, and K. Berberich, "Knowledge questions from knowledge graphs," in *Proc. ACM SIGIR Int. Conf. Theory of Inf. Retrieval*, Oct. 1-4, 2017, pp. 11–18.

[66] I. V. Serban, A. García-Durán, C. Gulcehre, S. Ahn, S. Chandar, A. Courville, and Y. Bengio, "Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus," 2016, *arXiv:1603.06807*.

[67] S. Reddy, D. Raghu, M. M. Khapra, and S. Joshi, "Generating natural language question-answer pairs from a knowledge graph using a rnn based question generation model," in *Proc. 15th Conf. Eur. Chap. Assoc. Comput. Ling.*, vol. 1, Apr. 3-7, 2017, pp. 376–385.

[68] H. Elsahar, C. Gravier, and F. Laforest, "Zero-shot question generation from knowledge graphs for unseen predicates and entity types," 2018, *arXiv:1802.06842*.

[69] V. Kumar, Y. Hua, G. Ramakrishnan, G. Qi, L. Gao, and Y.-F. Li, "Difficulty-controllable multi-hop question generation from knowledge graphs," in *Proc. 2019 Int. Semantic Web Conf.*, Oct. 26-30, 2019, pp. 382–398.

[70] J. Mostow and W. Chen, "Generating instruction automatically for the reading strategy of self-questioning," in *Proc. 14th Int. Conf. Artif. Intell. Educ.*, Jul. 6-10, 2009, pp. 465–472.

[71] M. Heilman and N. A. Smith, "Good question! statistical ranking for question generation," in *Proc. 2010 Annu. Conf. N. Amer. Chap. Assoc. Comput. Ling.*, Jun. 2-4, 2010, pp. 609–617.

[72] M. Heilman, "Automatic factual question generation from text," Ph.D. dissertation, Lang. Technol. Inst., Carnegie Mellon Univ., Pittsburgh, PA, USA, 2011.

[73] X. Du, J. Shao, and C. Cardie, "Learning to ask: Neural question generation for reading comprehension," 2017, *arXiv:1705.00106*.

[74] Q. Zhou, N. Yang, F. Wei, C. Tan, H. Bao, and M. Zhou, "Neural question generation from text: A preliminary study," in *Nat. CCF Conf. Natural Lang. Process. and Chin. Comput.*, Nov. 8-12, 2017, pp. 662–671.

[75] L. Song, Z. Wang, W. Hamza, Y. Zhang, and D. Gildea, "Leveraging context information for natural question generation," in *Proc. 2018 Conf. N. Amer. Chap. Assoc. Comput. Ling.: Human Lang. Technol.*, vol. 2, Jun. 1-6, 2018, pp. 569–574.

[76] V. Kumar, K. Boorla, Y. Meena, G. Ramakrishnan, and Y.-F. Li, "Automating reading comprehension by generating question and answer pairs," in *Pacific-Asia Conf. Knowl. Discovery and Data Mining*, Jun. 3-6, 2018, pp. 335–348.

[77] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances Neural Inf. Process. Syst.*, Dec. 8-13, 2014, pp. 3104–3112.

[78] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder–decoder for statistical machine translation," in *Proc. 2014 Conf. Empirical Meth. Natural Lang. Process.*, Oct. 25-29, 2014, pp. 1724–1734.

[79] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," 2015, *arXiv:1508.04025*.

[80] R. Nallapati, B. Zhou, C. Gulcehre, B. Xiang *et al.*, "Abstractive text summarization using sequence-to-sequence rnns and beyond," 2016, *arXiv:1602.06023*.

[81] R. Paulus, C. Xiong, and R. Socher, "A deep reinforced model for abstractive summarization," 2017, *arXiv:1705.04304*.

[82] I. V. Serban, A. Sordoni, Y. Bengio, A. C. Courville, and J. Pineau, "Building end-to-end dialogue systems using generative hierarchical neural network models," in *Proc. 30th AAAI Conf. Artif. Intell.*, vol. 16, Feb. 12-17, 2016, pp. 3776–3784.

[83] I. V. Serban, A. Sordoni, R. Lowe, L. Charlin, J. Pineau, A. Courville, and Y. Bengio, "A hierarchical latent variable encoder-decoder model for generating dialogues," in *Proc. 31st AAAI Conf. Artif. Intell.*, Feb. 4-9, 2017, pp. 3295–3301.

[84] V. Kumar, G. Ramakrishnan, and Y.-F. Li, "A framework for automatic question generation from text using deep reinforcement learning," 2018, *arXiv:1808.04961*.

[85] S. Subramanian, T. Wang, X. Yuan, S. Zhang, Y. Bengio, and A. Trischler, "Neural models for key phrase detection and question generation," 2017, *arXiv:1706.04560*.

[86] X. Yuan, T. Wang, C. Gulcehre, A. Sordoni, P. Bachman, S. Subramanian, S. Zhang, and A. Trischler, "Machine comprehension by text-to-text neural question generation," 2017, *arXiv:1705.02012*.

[87] Y. Zhao, X. Ni, Y. Ding, and Q. Ke, "Paragraph-level neural question generation with maxout pointer and gated self-attention networks," in *Proc. 2018 Conf. Empirical Meth. Natural Lang. Process.*, Oct. 31 - Nov. 4, 2018, pp. 3901–3910.

[88] B. Liu, M. Zhao, D. Niu, K. Lai, Y. He, H. Wei, and Y. Xu, "Learning to generate questions by learning what not to generate," 2019, *arXiv:1902.10418*.

[89] L. Song, Z. Wang, and W. Hamza, "A unified query-based generative model for question generation and question answering," 2017, *arXiv:1709.01058*.

[90] Y. Kim, H. Lee, J. Shin, and K. Jung, "Improving neural question generation using answer separation," 2018, *arXiv:1809.02393*.

[91] X. Sun, J. Liu, Y. Lyu, W. He, Y. Ma, and S. Wang, "Answer-focused and position-aware neural question generation," in *Proc. 2018 Conf. Empirical Meth. Natural Lang. Process.*, Aug. 28-30, 2018, pp. 3930–3939.

[92] D. Tang, N. Duan, T. Qin, Z. Yan, and M. Zhou, "Question answering and question generation as dual tasks," 2017, *arXiv:1706.02027*.

[93] T. Wang, X. Yuan, and A. Trischler, "A joint model for question answering and question generation," 2017, *arXiv:1706.01450*.

[94] X. Wang, B. Wang, T. Yao, Q. Zhang, and J. Xu, "Neural question generation with answer pivot," in *Proc. 34th AAAI Conf. Artif. Intell.*, Feb. 7-12, 2020, pp. 9138–9145.

[95] X. Du and C. Cardie, "Harvesting paragraph-level question-answer pairs from wikipedia," 2018, *arXiv:1805.05942*.

[96] V. Kumar, S. Muneeswaran, G. Ramakrishnan, and Y.-F. Li, "Paraqg: A system for generating questions and answers from paragraphs," 2019, *arXiv:1909.01642*.

[97] Y. Chen, L. Wu, and M. J. Zaki, "Bidirectional attentive memory networks for question answering over knowledge bases," in *Proc. 2019 Conf. N. Amer. Chap. Assoc. Comput. Ling.: Human Lang. Technol.*, vol. 1, Jun. 2-7, 2019, pp. 2913–2923.

[98] Y. Chen, L. Wu, and M. Zaki, "Graphflow: Exploiting conversation flow with graph neural networks for conversational machine comprehension," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Jul. 2020, pp. 1230–1236.

[99] S. Haussmann, O. Seneviratne, Y. Chen, Y. Ne'eman, J. Codella, C.-H. Chen, D. L. McGuinness, and M. J. Zaki, "Foodkg: A semantics-driven knowledge graph for food recommendation," in *Proc. 18th Int. Semantic Web Conf.*, vol. 11779, Oct. 26-30, 2019, pp. 146–162.

[100] S. Haussmann, Y. Chen, O. Seneviratne, N. Rastogi, J. Codella, C.-H. Chen, D. L. McGuinness, and M. J. Zaki, "Foodkg enabled q&a application," in *Proc. ISWC 2019 Satell. Tracks (Posters & Demonstrations, Industry, and Outrageous Ideas) co-located with 18th Int. Semantic Web Conf.*, vol. 2456, Oct. 26-30, 2019, pp. 273–276.

[101] S. Zhang and M. Bansal, "Addressing semantic drift in question generation for semi-supervised question answering," 2019, *arXiv:1909.06356*.

[102] A. R. Fabbri, P. Ng, Z. Wang, R. Nallapati, and B. Xiang, "Template-based question generation from retrieved sentences for improved unsupervised question answering," 2020, *arXiv:2004.11892*.

[103] J. Yu, X. Quan, Q. Su, and J. Yin, "Generating multi-hop reasoning questions to improve machine reading comprehension," in *Proc. 2020 Web Conf.*, Apr. 20-24, 2020, pp. 281–291.

[104] D. Lindberg, F. Popowich, J. Nesbit, and P. Winne, "Generating natural language questions to support learning on-line," in *Proc. 14th Eur. Workshop Natural Lang. Gener.*, Aug. 8-9, 2013, pp. 105–114.

[105] G. Danon and M. Last, "A syntactic approach to domain-specific automatic question generation," 2017, *arXiv:1712.09827*.

[106] L. A. Tuan, D. J. Shah, and R. Barzilay, "Capturing greater context for question generation," 2019, *arXiv:1910.10274*.

[107] V. Kumar, R. Chaki, S. T. Talluri, G. Ramakrishnan, Y.-F. Li, and G. Haffari, "Question generation from paragraphs: A tale of two hierarchical models," 2019, *arXiv:1911.03407*.

[108] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances Neural Inf. Process. Syst.*, Dec. 4-9, 2017, pp. 5998–6008.

[109] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, "Sequence level training with recurrent neural networks," 2015, *arXiv:1511.06732*.

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

[111] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, no. 3-4, pp. 229–256, May 1992.

[112] L. Pan, W. Lei, T.-S. Chua, and M.-Y. Kan, "Recent advances neural question generation," 2019, *arXiv:1905.08949*.

[113] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proc. 40th Annu. Meeting on Assoc. Comput. Ling.*, Jul. 6-12, 2002, pp. 311–318.

[114] S. Banerjee and A. Lavie, "Meteor: An automatic metric for mt evaluation with improved correlation with human judgments," in *Proc. Assoc. Comput. Ling. Workshop Intrinsic and Extrinsic Eval. Measures Mach. Transl. and/or Summarization*, Jun. 29, 2005, pp. 65–72.

[115] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*, Jul. 2004, pp. 74–81.

[116] C. Callison-Burch, M. Osborne, and P. Koehn, "Re-evaluation the role of bleu in machine translation research," in *Proc. 11th Conf. Eur. Chap. Assoc. Comput. Ling.*, Apr. 3-7, 2006, pp. 249–256.

[117] C.-W. Liu, R. Lowe, I. V. Serban, M. Noseworthy, L. Charlin, and J. Pineau, "How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation," 2016, *arXiv:1603.08023*.

[118] P. Nema and M. M. Khapra, "Towards a better metric for evaluating question generation systems," 2018, *arXiv:1808.10192*.

[119] W. Zhao, M. Peyrard, F. Liu, Y. Gao, C. M. Meyer, and S. Eger, "Moverscore: Text generation evaluating with contextualized embeddings and earth mover distance," 2019, *arXiv:1909.02622*.

[120] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with bert," 2019, *arXiv:1904.09675*.

[121] T. Sellam, D. Das, and A. P. Parikh, "Bleurt: Learning robust metrics for text generation," 2020, *arXiv:2004.04696*.

[122] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," 2018, *arXiv:1802.05365*.

[123] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.

[124] L. S. Zettlemoyer and M. Collins, "Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars," 2012, *arXiv:1207.1420*.

[125] Y. W. Wong and R. Mooney, "Learning synchronous grammars for semantic parsing with lambda calculus," in *Proc. 45th Annu. Meeting Assoc. Comput. Ling.*, Jun. 23-30, 2007, pp. 960–967.

[126] T. Kwiatkowski, L. Zettlemoyer, S. Goldwater, and M. Steedman, "Inducing probabilistic ccg grammars from logical form with higher-order unification," in *Proc. 2010 Conf. Empirical Meth. Natural Lang. Process.*, Oct. 9-11, 2010, pp. 1223–1233.

[127] C. Unger, L. Bühmann, J. Lehmann, A.-C. Ngonga Ngomo, D. Gerber, and P. Cimiano, "Template-based question answering over rdf data," in *Proc. 21st World Wide Web Conf.*, Apr. 16-20, 2012, pp. 639–648.

[128] J. Berant and P. Liang, "Imitation learning of agenda-based semantic parsers," *Trans. Assoc. Comput. Ling.*, vol. 3, pp. 545–558, Nov. 2015.

[129] S. Reddy, O. Täckström, M. Collins, T. Kwiatkowski, D. Das, M. Steedman, and M. Lapata, "Transforming dependency structures to logical forms for semantic parsing," *Trans. Assoc. Comput. Ling.*, vol. 4, pp. 127–140, Apr. 2016.

[130] J. Bao, N. Duan, Z. Yan, M. Zhou, and T. Zhao, "Constraint-based question answering with knowledge graph," in *26th Int. Conf. Comput. Ling.*, Dec. 11-16, 2016, pp. 2503–2514.

[131] A. Abujabal, M. Yahya, M. Riedewald, and G. Weikum, "Automated template generation for question answering over knowledge graphs," in *Proc. 26th Int. Conf. World Wide Web*, Apr. 3-7, 2017, pp. 1191–1200.

[132] S. Hu, L. Zou, J. X. Yu, H. Wang, and D. Zhao, "Answering natural language questions by subgraph matching over knowledge graphs," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 5, pp. 824–837, Oct. 2018.

[133] H. Bast and E. Haussmann, "More accurate question answering on freebase," in *Proc. 24th ACM Int. Conf. Inf. and Knowl. Manage.*, Oct. 19-23, 2015, pp. 1431–1440.

[134] S. W.-t. Yih, M.-W. Chang, X. He, and J. Gao, "Semantic parsing via staged query graph generation: Question answering with knowledge base," in *Proc. 53rd Annu. Meeting Assoc. Comput. Ling.*, Jul. 26-31, 2015, pp. 1321–1331.

[135] Q. Cai and A. Yates, "Large-scale semantic parsing via schema matching and lexicon extension," in *Proc. 51st Annu. Meeting Assoc. Comput. Ling.*, vol. 1, Aug. 4-9, 2013, pp. 423–433.

[136] J. Krishnamurthy and T. M. Mitchell, "Weakly supervised training of semantic parsers," in *Proc. 2012 Joint Conf. Empirical Meth. Natural Lang. Process. and Comput. Natural Lang. Learn.*, Jul. 12-14, 2012, pp. 754–765.

[137] X. Yao and B. Van Durme, "Information extraction over structured data: Question answering with freebase," in *Proc. 52nd Annu. Meeting Assoc. Comput. Ling.*, vol. 1, Jun. 22-27, 2014, pp. 956–966.

[138] S. Yavuz, I. Gur, Y. Su, M. Srivatsa, and X. Yan, "Improving semantic parsing via answer type inference," in *Proc. 2016 Conf. Empirical Meth. Natural Lang. Process.*, Nov. 1-4, 2016, pp. 149–159.

[139] K. J. Han, A. Chandrashekaran, J. Kim, and I. Lane, "The capio 2017 conversational speech recognition system," 2017, *arXiv:1801.00059*.

[140] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, *arXiv:1409.0473*.

[141] C. Tan, F. Wei, N. Yang, W. Lv, and M. Zhou, "S-net: From answer extraction to answer generation for machine reading comprehension," 2017, *arXiv:1706.04815*.

[142] Y. Feng, S. Huang, D. Zhao *et al.*, "Hybrid question answering over knowledge base and free text," in *Proc. 26th Int. Conf. Comput. Ling.*, Dec. 11-16, 2016, pp. 2397–2407.

[143] J. Weston, S. Chopra, and A. Bordes, "Memory networks," 2014, *arXiv:1410.3916*.

[144] W. W. Cohen, "Tensorlog: A differentiable deductive database," 2016, *arXiv:1605.06523*.

[145] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, "Teaching machines to read and comprehend," in *Advances Neural Inf. Process. Syst.*, Dec. 7-12, 2015, pp. 1693–1701.

[146] Y. Cui, Z. Chen, S. Wei, S. Wang, T. Liu, and G. Hu, "Attention-over-attention neural networks for reading comprehension," 2016, *arXiv:1607.04423*.

[147] C. Xiong, V. Zhong, and R. Socher, "Dynamic coattention networks for question answering," 2016, *arXiv:1611.01604*.

[148] S. Kundu and H. T. Ng, "A question-focused multi-factor attention network for answering," 2018, *arXiv:1801.08290*.

[149] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," 2016, *arXiv:1606.05250*.

[150] C. Zhu, M. Zeng, and X. Huang, "Sdnet: Contextualized attention-based deep network for conversational question answering," 2018, *arXiv:1812.03593*.

[151] H.-Y. Huang, E. Choi, and W.-t. Yih, "Flowqa: Grasping flow in history for conversational machine comprehension," 2018, *arXiv:1810.06683*.

[152] Y. Chen, L. Wu, and M. J. Zaki, "Reinforcement learning based graph-to-sequence model for natural question generation," in *Proc. 8th Int. Conf. Learn. Representations*, Apr. 26-30, 2020.

[153] L. Pan, Y. Xie, Y. Feng, T.-S. Chua, and M.-Y. Kan, "Semantic graphs for generating deep questions," 2020, *arXiv:2004.12704*.

[154] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Advances Neural Inf. Process. Syst.*, Dec. 7-12, 2015, pp. 2692–2700.

[155] J. Gu, Z. Lu, H. Li, and V. O. Li, "Incorporating copying mechanism in sequence-to-sequence learning," 2016, *arXiv:1603.06393*.

[156] Z. Tu, Z. Lu, Y. Liu, X. Liu, and H. Li, "Modeling coverage for neural machine translation," 2016, *arXiv:1601.04811*.

[157] L. Dong, N. Yang, W. Wang, F. Wei, X. Liu, Y. Wang, J. Gao, M. Zhou, and H.-W. Hon, "Unified language model pre-training for natural language understanding and generation," in *Advances Neural Inf. Process. Syst.*, Dec. 8-14, 2019, pp. 13 042–13 054.

[158] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[159] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[160] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. 2016 IEEE Conf. Comput. Vision and Pattern Recognit.*, Jun. 27-30, 2016, pp. 770–778.

[161] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *IEEE Int. Conf. Acoust., Speech and Signal Process.*, May 26-31, 2013, pp. 6645–6649.

[162] Y. Chen and M. J. Zaki, "KATE: k-competitive autoencoder for text," in *Proc. 23rd Int. Conf. Knowl. Discovery and Data Mining*, Aug. 13-17, 2017, pp. 85–94.

[163] J. You, B. Liu, Z. Ying, V. Pande, and J. Leskovec, "Graph convolutional policy network for goal-directed molecular graph generation," in *Advances Neural Inf. Process. Syst.*, Dec. 3-8, 2018, pp. 6410–6421.

[164] Y. Chen, R. M. Rabbani, A. Gupta, and M. J. Zaki, "Comparative text analytics via topic modeling in banking," in *Proc. 2017 IEEE Symp. Series Comput. Intell.*, Nov. 27 - Dec. 1, 2017, pp. 1–8.

[165] J. Zou, M. Huss, A. Abid, P. Mohammadi, A. Torkamani, and A. Telenti, "A primer on deep learning in genomics," *Nature Genetics*, vol. 51, no. 1, pp. 12–18, Nov. 2019.

[166] S. Liu, Y. Chen, X. Xie, J. K. Siow, and Y. Liu, "Automatic code summarization via multi-dimensional semantic fusing in gnn," 2020, *arXiv:2006.05405*.

[167] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. of Artif. Intell. Res.*, vol. 4, pp. 237–285, May 1996.

[168] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[169] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ. San Diego, Tech. Rep., 1985.

[170] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Networks*, vol. 5, no. 2, pp. 157–166, Mar. 1994.

[171] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," in *A Field Guide to Dynamical Recurrent Neural Networks*. Piscataway, NJ, USA: IEEE Press, 2001, ch. 14, pp. 237–244.

[172] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[173] J. Koutnik, K. Greff, F. Gomez, and J. Schmidhuber, "A clockwork rnn," 2014, *arXiv:1402.3511*.

[174] K. Yao, T. Cohn, K. Vylomova, K. Duh, and C. Dyer, "Depth-gated lstm," 2015, *arXiv:1508.03790*.

[175] G.-B. Zhou, J. Wu, C.-L. Zhang, and Z.-H. Zhou, "Minimal gated unit for recurrent neural networks," *Int. J. of Automat. and Comput.*, vol. 13, no. 3, pp. 226–234, Jun. 2016.

[176] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *Int. Conf. Mach. Learn.*, Jul. 6-11, 2015, pp. 2342–2350.

[177] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE Trans. on Neural Networks and Learn. Syst.*, vol. 28, no. 10, pp. 2222–2232, Mar. 2016.

[178] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. on Signal Process.*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.

[179] S. Fernández, A. Graves, and J. Schmidhuber, "An application of recurrent neural networks to discriminative keyword spotting," in *Int. Conf. Artif. Neural Networks*, Sep. 9-13, 2007, pp. 220–229.

[180] N. Boulanger-Lewandowski, "Modeling high-dimensional audio sequences with recurrent neural networks," Ph.D. dissertation, Dept. Comput. Sci. and Oper. Res., Univ. Montreal, Montreal, QC, CA, 2014.

[181] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *IEEE Trans. on Pattern Anal. and Mach. Intell.*, vol. 31, no. 5, pp. 855–868, May 2008.

[182] D. Golub and X. He, "Character-level question answering with attention," 2016, *arXiv:1604.00727*.

[183] W. Yin, M. Yu, B. Xiang, B. Zhou, and H. Schütze, "Simple question answering by attentive convolutional neural network," 2016, *arXiv:1606.03391*.

[184] R. Kadlec, M. Schmid, O. Bajgar, and J. Kleindienst, "Text understanding with the attention sum reader network," 2016, *arXiv:1603.01547*.

[185] A. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston, "Key-value memory networks for directly reading documents," 2016, *arXiv:1606.03126*.

[186] J. Yin, X. Jiang, Z. Lu, L. Shang, H. Li, and X. Li, "Neural generative question answering," 2015, *arXiv:1512.01337*.

[187] M. Eric and C. D. Manning, "Key-value retrieval networks for task-oriented dialogue," 2017, *arXiv:1705.05414*.

[188] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proc. 20th Int. Conf. Knowl. Discovery and Data Mining*, Aug. 24-27, 2014, pp. 701–710.

[189] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. 22nd Int. Conf. Knowl. Discovery and Data Mining*, Aug. 13-17, 2016, pp. 855–864.

[190] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proc. 24th Int. Conf. World Wide Web*, May 18-22, 2015, pp. 1067–1077.

[191] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Chang, "Network representation learning with rich text information," in *Proc. 24th Int. Joint Conf. Artif. Intell.*, Jul. 25-31, 2015, pp. 2111–2117.

[192] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. on Neural Networks*, vol. 20, no. 1, pp. 61–80, Dec. 2008.

[193] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.

[194] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Process. Mag.*, vol. 34, no. 4, pp. 18–42, Jul. 2017.

[195] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, Aug. 2017, pp. 1263–1272.

[196] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances Neural Inf. Process. Syst.*, Dec. 4-9, 2017, pp. 1024–1034.

[197] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," 2015, *arXiv:1511.05493*.

[198] K. Xu, L. Wu, Z. Wang, and V. Sheinin, "Graph2seq: Graph to sequence learning with attention-based neural networks," 2018, *arXiv:1804.00823*.

[199] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner *et al.*, "Relational inductive biases, deep learning, and graph networks," 2018, *arXiv:1806.01261*.

[200] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," 2013, *arXiv:1312.6203*.

[201] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," 2015, *arXiv:1506.05163*.

[202] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *Advances Neural Inf. Process. Syst.*, Dec. 7-12, 2015, pp. 2224–2232.

[203] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances Neural Inf. Process. Syst.*, Dec. 5-10, 2016, pp. 3844–3852.

[204] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," 2017, *arXiv:1710.10903*.

[205] J. Zhang, X. Shi, J. Xie, H. Ma, I. King, and D.-Y. Yeung, "Gaan: Gated attention networks for learning on large and spatiotemporal graphs," 2018, *arXiv:1803.07294*.

[206] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *Advances Neural Inf. Process. Syst.*, Dec. 3-8, 2018, pp. 4800–4810.

[207] H. Gao and S. Ji, "Graph u-nets," 2019, *arXiv:1905.05178*.

[208] D. Teney, L. Liu, and A. van den Hengel, "Graph-structured representations for visual question answering," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognit.*, Jul. 21-26, 2017, pp. 1–9.

[209] W. Norcliffe-Brown, S. Vafeias, and S. Parisot, "Learning conditioned graph structures for interpretable visual question answering," in *Advances Neural Inf. Process. Syst.*, Dec. 3-8, 2018, pp. 8344–8353.

[210] Z. Wang, T. Chen, J. Ren, W. Yu, H. Cheng, and L. Lin, "Deep reasoning with knowledge graph for social relationship understanding," 2018, *arXiv:1807.00504*.

[211] M. Narasimhan, S. Lazebnik, and A. Schwing, "Out of the box: Reasoning with graph convolution nets for factual visual question answering," in *Advances Neural Inf. Process. Syst.*, Dec. 3-8, 2018, pp. 2659–2670.

[212] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proc. 24th Int. Conf. Knowl. Discovery and Data Mining*, Aug. 19-23, 2018, pp. 974–983.

[213] C. Xu, P. Zhao, Y. Liu, V. S. Sheng, J. Xu, F. Zhuang, J. Fang, and X. Zhou, "Graph contextualized self-attention network for session-based recommendation," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 10-16, 2019, pp. 3940–3946.

[214] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph neural networks for social recommendation," in *Proc. World Wide Web Conf.*, May 13-17, 2019, pp. 417–426.

[215] H. Wang, F. Zhang, M. Zhang, J. Leskovec, M. Zhao, W. Li, and Z. Wang, "Knowledge-aware graph neural networks with label smoothness regularization for recommender

systems," in *Proc. 25th Int. Conf. Knowl. Discovery and Data Mining*, Aug. 4-8, 2019, pp. 968–977.

[216] R. Yin, K. Li, G. Zhang, and J. Lu, "A deeper graph neural network for recommender systems," *Knowl.-Based Syst.*, vol. 185, p. 105020, Dec. 2019.

[217] A. Deac, Y.-H. Huang, P. Veličković, P. Liò, and J. Tang, "Drug-drug adverse effect prediction with graph co-attention," 2019, *arXiv:1905.00534*.

[218] T. Nguyen, H. Le, and S. Venkatesh, "Graphdta: prediction of drug–target binding affinity using graph convolutional networks," *BioRxiv*, p. 684662, Jul. 2019.

[219] W. Torng and R. B. Altman, "Graph convolutional neural networks for predicting drug-target interactions," *J. of Chem. Inf. and Model.*, vol. 59, no. 10, pp. 4131–4149, Oct. 2019.

[220] M. Sun, S. Zhao, C. Gilvary, O. Elemento, J. Zhou, and F. Wang, "Graph convolutional networks for computational drug development and discovery," *Briefings in Bioinf.*, vol. 21, no. 3, pp. 919–935, May 2020.

[221] B. Tang, S. T. Kramer, M. Fang, Y. Qiu, Z. Wu, and D. Xu, "A self-attention based message passing neural network for predicting molecular lipophilicity and aqueous solubility," *J. Cheminformatics*, vol. 12, no. 1, pp. 1–9, Feb. 2020.

[222] C. Shi, M. Xu, Z. Zhu, W. Zhang, M. Zhang, and J. Tang, "Graphaf: a flow-based autoregressive model for molecular graph generation," 2020, *arXiv:2001.09382*.

[223] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, Feb. 7-12, 2019, pp. 7370–7377.

[224] H. Sun, B. Dhingra, M. Zaheer, K. Mazaitis, R. Salakhutdinov, and W. W. Cohen, "Open domain question answering using early fusion of knowledge bases and text," 2018, *arXiv:1809.00782*.

[225] N. De Cao, W. Aziz, and I. Titov, "Question answering by reasoning across documents with graph convolutional networks," 2018, *arXiv:1808.09920*.

[226] L. Song, Z. Wang, M. Yu, Y. Zhang, R. Florian, and D. Gildea, "Exploring graph-structured passage representation for multi-hop reading comprehension with graph neural networks," 2018, *arXiv:1809.02040*.

[227] K. Xu, L. Wu, Z. Wang, M. Yu, L. Chen, and V. Sheinin, "Sql-to-text generation with graph-to-sequence model," 2018, *arXiv:1809.05255*.

[228] W. Hu, Z. Chan, B. Liu, D. Zhao, J. Ma, and R. Yan, "Gsn: A graph-structured network for multi-party dialogues," 2019, *arXiv:1905.13637*.

[229] D. Ghosal, N. Majumder, S. Poria, N. Chhaya, and A. Gelbukh, "Dialoguegcn: A graph convolutional neural network for emotion recognition in conversation," 2019, *arXiv:1908.11540*.

[230] J. Bastings, I. Titov, W. Aziz, D. Marcheggiani, and K. Sima'an, "Graph convolutional encoders for syntax-aware neural machine translation," 2017, *arXiv:1704.04675*.

[231] D. Beck, G. Haffari, and T. Cohn, "Graph-to-sequence learning using gated graph neural networks," 2018, *arXiv:1806.09835*.

[232] K. Xu, L. Wu, Z. Wang, M. Yu, L. Chen, and V. Sheinin, "Exploiting rich syntactic information for semantic parsing with graph-to-sequence model," 2018, *arXiv:1808.07624*.

[233] L. Song, Y. Zhang, Z. Wang, and D. Gildea, "A graph-to-sequence model for amr-to-text generation," 2018, *arXiv:1805.02473*.

[234] D. Marcheggiani and L. Perez-Beltrachini, "Deep graph convolutional encoders for structured data to text generation," 2018, *arXiv:1810.09995*.

[235] B. Distiawan, J. Qi, R. Zhang, and W. Wang, "Gtr-lstm: A triple encoder for sentence generation from rdf data," in *Proc. 56th Annu. Meeting Assoc. Comput. Ling.*, vol. 1, Jul. 15-20, 2018, pp. 1627–1637.

[236] P. Vougiouklis, H. Elsahar, L.-A. Kaffee, C. Gravier, F. Laforest, J. Hare, and E. Simperl, "Neural wikipedian: Generating textual summaries from knowledge base triples," *J. of Web Semantics*, vol. 52, pp. 1–15, Oct. 2018.

[237] Y. Chen, L. Wu, and M. J. Zaki, "Toward subgraph guided knowledge graph question generation with graph neural networks," 2020, *arXiv:2004.06015*.

[238] P. Liu, S. Chang, X. Huang, J. Tang, and J. C. K. Cheung, "Contextualized non-local neural networks for sequence learning," 2018, *arXiv:1811.08600*.

[239] Y. Chen, L. Wu, and M. J. Zaki, "Iterative deep graph learning for graph neural networks: Better and robust node embeddings," 2020, *arXiv:2006.13009*.

[240] L. Franceschi, M. Niepert, M. Pontil, and X. He, "Learning discrete structures for graph neural networks," 2019, *arXiv:1903.11960*.

[241] Wikipedia contributors, "Reinforcement learning — Wikipedia, the free encyclopedia," https://en.wikipedia.org/w/index.php?title=Reinforcement_learning&oldid=962012410 (Accessed Jun. 11, 2020).

[242] M. Kearns and S. Singh, "Near-optimal reinforcement learning in polynomial time," *Mach. Learn.*, vol. 49, no. 2-3, pp. 209–232, Nov. 2002.

[243] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," 2017, *arXiv:1708.05866*.

[244] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," 2018, *arXiv:1811.12560*.

[245] C. J. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3-4, pp. 279–292, May 1992.

[246] G. A. Rummery and M. Niranjan, *On-Line Q-Learning Using Connectionist Systems,* vol. 37. Cambridge, U.K.: Univ. Cambridge, 1994.

[247] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.

[248] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971* .

[249] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[250] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," 2017, *arXiv:1712.01815*.

[251] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017.

[252] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel *et al.*, "Mastering atari, go, chess and shogi by planning with a learned model," 2019, *arXiv:1911.08265*.

[253] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, Oct. 2019.

[254] N. Kohl and P. Stone, "Policy gradient reinforcement learning for fast quadrupedal locomotion," in *IEEE Int. Conf. Robot. and Automat.*, vol. 3, Apr. 26 - May 1, 2004, pp. 2619–2624.

[255] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, no. 7553, pp. 503–507, May 2015.

[256] K. Chatzilygeroudis, R. Rama, R. Kaushik, D. Goepp, V. Vassiliades, and J.-B. Mouret, "Black-box data-efficient policy search for robotics," in *Proc. 2017 IEEE/RSJ Int. Conf. Intell. Robots and Syst.*, Sep. 24-28, 2017, pp. 51–58.

[257] P.-H. Su, M. Gasic, N. Mrksic, L. Rojas-Barahona, S. Ultes, D. Vandyke, T.-H. Wen, and S. Young, "On-line active reward learning for policy optimisation in spoken dialogue systems," 2016, *arXiv:1605.07669*.

[258] B. Dhingra, L. Li, X. Li, J. Gao, Y.-N. Chen, F. Ahmed, and L. Deng, "Towards end-to-end reinforcement learning of dialogue agents for information access," 2016, *arXiv:1609.00777*.

[259] J. Li, W. Monroe, A. Ritter, M. Galley, J. Gao, and D. Jurafsky, "Deep reinforcement learning for dialogue generation," 2016, *arXiv:1606.01541*.

[260] J. C. Caicedo and S. Lazebnik, "Active object localization with deep reinforcement learning," in *Proc. 2015 IEEE Int. Conf. Comput. Vision*, Dec. 7-13, 2015, pp. 2488–2496.

[261] Z. Jie, X. Liang, J. Feng, X. Jin, W. Lu, and S. Yan, "Tree-structured reinforcement learning for sequential object localization," in *Advances Neural Inf. Process. Syst.*, Dec. 5-10, 2016, pp. 127–135.

[262] S. Mathe, A. Pirinen, and C. Sminchisescu, "Reinforcement learning for visual object detection," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognit.*, Jun. 27-30, 2016, pp. 2894–2902.

[263] Y. Ling, S. A. Hasan, V. Datla, A. Qadir, K. Lee, J. Liu, and O. Farri, "Diagnostic inferencing via improving clinical concept extraction with deep reinforcement learning: A preliminary study," in *Mach. Learn. for Healthcare Conf.*, Aug. 18-19, 2017, pp. 271–285.

[264] Y. Liu, O. Gottesman, A. Raghu, M. Komorowski, A. A. Faisal, F. Doshi-Velez, and E. Brunskill, "Representation balancing mdps for off-policy policy evaluation," in *Advances Neural Inf. Process. Syst.*, Dec. 3-8, 2018, pp. 2644–2653.

[265] N. Kallus and A. Zhou, "Confounding-robust policy improvement," in *Advances Neural Inf. Process. Syst.*, Dec. 3-8, 2018, pp. 9269–9279.

[266] Y. Gao, L. Wu, H. Homayoun, and L. Zhao, "Dyngraph2seq: Dynamic-graph-to-sequence interpretable learning for health stage prediction in online health forums," 2019, *arXiv:1908.08497*.

[267] A. W. Senior, R. Evans, J. Jumper, J. Kirkpatrick, L. Sifre, T. Green, C. Qin, A. Žídek, A. W. Nelson, A. Bridgland *et al.*, "Improved protein structure prediction using potentials from deep learning," *Nature*, vol. 577, no. 7792, pp. 706–710, Jan. 2020.

[268] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015, *arXiv:1502.03167*.

[269] Y. Yang and M.-W. Chang, "S-mart: Novel tree-based structured learning algorithms applied to tweet entity linking," 2016, *arXiv:1609.08075*.

[270] Z. Wang, S. Yan, H. Wang, and X. Huang, "An overview of microsoft deep qa system on stanford webquestions benchmark," Microsoft Res., Tech. Rep. MSR-TR-2014-121, 2014.

[271] S. Reddy, O. Täckström, S. Petrov, M. Steedman, and M. Lapata, "Universal semantic parsing," in *Proc. 2017 Conf. Empirical Meth. Natural Lang. Process.*, Sep. 9-11, 2017, pp. 89–101.

[272] P. Baudis and J. Pichl, "Dataset factoid webquestions," https://github.com/brmson/dataset-factoid-webquestions (Accessed May 4, 2016).

[273] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proc. 2014 Conf. Empirical Meth. Natural Lang. Process.*, Oct. 25-29, 2014, pp. 1532–1543.

[274] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[275] K. Lee, S. Salant, T. Kwiatkowski, A. Parikh, D. Das, and J. Berant, "Learning recurrent span representations for extractive question answering," 2016, *arXiv:1611.01436*.

[276] W. Wang, M. Yan, and C. Wu, "Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering," 2018, *arXiv:1811.11934*.

[277] C. Clark and M. Gardner, "Simple and effective multi-paragraph reading comprehension," 2017, *arXiv:1710.10723*.

[278] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances Neural Inf. Process. Syst.*, Dec. 5-8, 2013, pp. 2787–2795.

[279] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognit.*, Jul. 21-26, 2017, pp. 3693–3702.

[280] F. W. Levi, *Finite Geometrical Systems: Six Public Lectures.* Kolkata, India: Univ. Calcutta, 1942.

[281] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn.*, Jun. 21-24, 2010, pp. 807–814.

[282] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Advances Neural Inf. Process. Syst.*, Dec. 7-12, 2015, pp. 1171–1179.

[283] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel, "Self-critical sequence training for image captioning," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognit.*, Jul. 21-26, 2017, pp. 7008–7024.

[284] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. Rush, "OpenNMT: Open-source toolkit for neural machine translation," in *Proc. 55th Annu. Meeting Assoc. Comput. Ling., Syst. Demonstrations*, Jul. 30 - Aug. 4, 2017, pp. 67–72.

[285] W.-t. Yih, M. Richardson, C. Meek, M.-W. Chang, and J. Suh, "The value of semantic parse labeling for knowledge base question answering," in *Proc. 54th Annu. Meeting Assoc. Comput. Ling.*, vol. 2, Aug. 7-12, 2016, pp. 201–206.

[286] A. Talmor and J. Berant, "The web as a knowledge-base for answering complex questions," 2018, *arXiv:1803.06643*.

[287] M. Zhou, M. Huang, and X. Zhu, "An interpretable reasoning network for multi-relation question answering," 2018, *arXiv:1801.04726*.

[288] D. P. Kingma, T. Salimans, and M. Welling, "Variational dropout and the local reparameterization trick," in *Advances Neural Inf. Process. Syst.*, Dec. 7-12, 2015, pp. 2575–2583.

[289] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger, "From word embeddings to document distances," in *Int. Conf. Mach. Learn.*, Jul. 6-11, 2015, pp. 957–966.

[290] H. Gong, S. Bhat, L. Wu, J. Xiong, and W.-m. Hwu, "Reinforcement learning based text style transfer without parallel training corpus," 2019, *arXiv:1903.10671.*

[291] Q. Zhou, N. Yang, F. Wei, and M. Zhou, "Sequential copying networks," in *Proc. 32nd AAAI Conf. Artif. Intell.*, Feb. 2-7, 2018, pp. 4987–4994.

[292] S. Sheng, L. Van Gool, and M. F. Moens, "A dataset for multimodal question answering in the cultural heritage domain," in *Proc. Workshop on Lang. Technol. Resour. and Tools for Digit. Humanities*, Dec. 11-17, 2016, pp. 10–17.

[293] A. Kembhavi, M. Seo, D. Schwenk, J. Choi, A. Farhadi, and H. Hajishirzi, "Are you smarter than a sixth grader? textbook question answering for multimodal machine comprehension," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognit.*, Jul. 21-26, 2017, pp. 4999–5007.

[294] H. Li, Y. Wang, G. De Melo, C. Tu, and B. Chen, "Multimodal question answering over structured data with ambiguous entities," in *Proc. 26th Int. Conf. World Wide Web Companion*, Apr. 3-7, 2017, pp. 79–88.

[295] B. Pan, H. Li, Z. Yao, D. Cai, and H. Sun, "Reinforced dynamic reasoning for conversational question generation," 2019, *arXiv:1907.12667.*

[296] M. Nakanishi, T. Kobayashi, and Y. Hayashi, "Towards answer-unaware conversational question generation," in *Proc. 2nd Workshop on Mach. Reading for Question Answering*, Nov. 4, 2019, pp. 63–71.

[297] N. Duan, D. Tang, P. Chen, and M. Zhou, "Question generation for question answering," in *Proc. 2017 Conf. Empirical Meth. Natural Lang. Process.*, Sep. 2017, pp. 866–874.

[298] Z. Yang, J. Hu, R. Salakhutdinov, and W. W. Cohen, "Semi-supervised qa with generative domain-adaptive nets," 2017, *arXiv:1702.02206*.

[299] D. Golub, P.-S. Huang, X. He, and L. Deng, "Two-stage synthesis networks for transfer learning in machine comprehension," 2017, *arXiv:1706.09789*.

[300] D. Tang, N. Duan, Z. Yan, Z. Zhang, Y. Sun, S. Liu, Y. Lv, and M. Zhou, "Learning to collaborate for question answering and asking," in *Proc. 2018 Conf. N. Amer. Chap. Assoc. Comput. Ling.: Human Lang. Technol.*, vol. 1, Jun. 1-6, 2018, pp. 1564–1574.

[301] M. Sachan and E. Xing, "Self-training for jointly learning to ask and answer questions," in *Proc. 2018 Conf. N. Amer. Chap. Assoc. Comput. Ling.: Human Lang. Technol.*, vol. 1, Jun. 1-6, 2018, pp. 629–640.

[302] H. Xiao, F. Wang, J. Yan, and J. Zheng, "Dual ask-answer network for machine reading comprehension," 2018, *arXiv:1809.01997*.

APPENDIX A
PERMISSIONS

Some figures used in this dissertation are from the following sources:

1. Fig. 2.1 is from `https://commons.wikimedia.org/wiki/File:The_LSTM_cell.png`, licensed under version 4.0 of the Creative Commons CC-BY license (CC BY 4.0).

2. Fig. 2.2 is from [185], licensed under version 4.0 of the Creative Commons CC-BY license (CC BY 4.0).

3. Fig. 2.4 is from [168], licensed under Attribution-NonCommercial-NoDerivs 2.0 Generic license (CC BY-NC-ND 2.0).