MONASH University

Complex Question Answering over Large-scale Knowledge Bases

Yuncheng Hua

Doctor of Philosophy

A thesis submitted for the degree of
Doctor of Philosophy (4319)
(JOINT PHD WITH SOUTHEAST UNIVERSITY - INTERNATIONAL) at

Monash University in *2021*

Clayton School of Information Technology

Copyright notice

Abstract

Many large-scale knowledge bases have been created to collect and organize factual knowledge. As KBs' scale expands, the effective retrieval of useful information from KBs has become an important problem. Knowledge Base Question Answering (KBQA), a method that automatically answers natural language questions (NLQ) raised by users, has become a natural way to understand users' intentions and retrieve relevant knowledge in KBs. To make KBQA more applicable in real-life settings, researchers have turned attention from simple questions into complex questions to meet users' increasingly complex needs. Compared with simple questions—a type of questions that refer to a single fact stored in a KB, complex questions often require logical, quantitative, and comparative reasoning over a series of KB triples. Therefore, Complex question-answering (CQA) has been a research hot-spot that attracts significant attention lately.

A compelling approach to CQA is interpreting the natural-language questions as a sequence of actions, which can then be directly executed on the KB to yield the answer, aka the neural program induction (NPI) approach. Such a programmer-interpreter approach, however, suffers from the sparse reward and the data inefficiency problems. The NPI approach also exhibits uneven performance when the questions have different types, harboring inherently different characteristics, e.g., level of difficulty.

This research proposes a complex question answering framework that is effective across a range of questions using a modest number of training samples. Our framework consists of a *neural generator* that transforms a NLQ into a sequence of primitive actions. Given a generated sequence, a *symbolic executor* executes the corresponding query on the knowledge graph to yield the answer. We equip our model with a memory buffer that stores high-reward promising programs to mitigate the sparse reward problem. Also, we encourage the model to

explore unseen space while keeping the past promising trials in the memory to improve the data efficiency. We propose a meta-reinforcement learning framework for program induction to tackle the potential distributional bias in questions to be answered. Our method could quickly and effectively adapt the meta-learned programmer to new questions based on the most similar questions retrieved from the training data. To find the most similar questions, we present a novel method that automatically learns a retrieval model alternately with the programmer from the weak supervision, i.e., the system's performance with respect to the produced answers. To validate the system's effectiveness, we conducted experiments on two datasets: CQA, a recent large-scale complex question answering dataset, and WebQuestionsSP, a multi-hop question answering dataset. On both datasets, our model outperformed the state-of-the-art models. Notably, on CQA, our model achieved better performance on the questions with higher complexity while only using approximately 1% of the total training samples.

Publications During Enrolment

1. Hua, Y., Li, Y., Qi, G., Wu, W., Zhang, J. and Qi, D. Less is more: Data-efficient complex question answering over knowledge bases. In Journal of Web Semantics, 2020, 65, p.100612.

2. Hua, Y., Li, Y., Haffari, G., Qi, G. and Wu, T. Few-shot Complex Knowledge Base Question Answering via Meta Reinforcement Learning. In *2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*, November 16-20, 2020, Proceedings, pages 5827-5837.

3. Hua, Y., Li, Y., Haffari, G., Qi, G. and Wu, W. Retrieve, Program, Repeat: Complex Knowledge Base Question Answering via Alternate Meta-learning. In *Twenty-Ninth International Joint Conference on Artificial Intelligence and Seventeenth Pacific Rim International Conference on Artificial Intelligence, IJCAI-PRICAI-20*, Yokohama, Japan, January 7-15, 2021, Proceedings, pages 3679-3686.

Thesis Including Published Works Declaration

In accordance with Monash University Doctorate Regulation 17.2 Doctor of Philosophy and Research Master's regulations the following declarations are made:

I hereby declare that this thesis contains no material which has been accepted for the award of any other degree or diploma at any university or equivalent institution and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

This thesis includes three original papers published in peer reviewed journals and conferences. The core theme of the thesis is 'Complex Question Answering over Large-scale Knowledge Bases'. The ideas, development and writing up of all the papers in the thesis were the principal responsibility of myself, the student, working within the Faculty of Information Technology and Monash University-Southeast University Joint Research Institute under the supervisions of Dr. Yuan-Fang Li and Prof. Guilin Qi.

The inclusion of co-authors reflects the fact that the work came from active collaboration between researchers and acknowledges input into team-based research.

In the case of Chapter 3, Chapter 4 and Chapter 5, my contribution to the work involved the following:

Thesis Chapter	Publication Title	Status (*published, in press, accepted or returned for revision, submitted*)	Nature and % of student contribution	Co-author name(s) Nature and % of Co-author's contribution*	Co-author(s), Monash student Y/N*
3	Less is more: Data-efficient complex question answering over knowledge bases	*Published*	Conceptualisation, implementing models, conducting experiments, formal analysis, investigation and manuscript writing, 70%	1. Yuan-Fang Li, Conceptualisation, supervision and manuscript writing, 10% 2. Guilin Qi, Conceptualisation, supervision and manuscript writing, 5% 3. Wei Wu, Implementing models and conducting experiments, 5% 4. Jingyao Zhang, Implementing models, 5% 5. Daiqing Qi, Conducting experiments, 5%	1. No 2. No 3. No 4. No 5. No
4	Few-shot Complex Knowledge Base Question Answering via Meta Reinforcement Learning	*Published*	Conceptualisation, implementing models, conducting experiments, formal analysis, investigation and manuscript writing, 70%	1. Yuan-Fang Li, Conceptualisation, supervision and manuscript writing, 15% 2. Gholamreza Haffari, Conceptualisation, 5% 3. Guilin Qi, Conceptualisation and supervision, 5% 4. Tongtong Wu, Conceptualisation, 5%	1. No 2. No 3. No 4. No

5	Retrieve, Program, Repeat: Complex Knowledge Base Question Answering via Alternate Meta-learning	*Published*	Conceptualisation, implementing models, conducting experiments, formal analysis, investigation and manuscript writing, 70%	1. Yuan-Fang Li, Conceptualisation, supervision and manuscript writing, 10% 2. Gholamreza Haffari, Conceptualisation and manuscript writing, 10% 3. Guilin Qi, Conceptualisation and supervision 5% 4. Wei Wu, Implementing models and conducting experiments, 5%	1. No 2. No 3. No 4. No

I have not renumbered the sections of submitted or published papers in order to generate a consistent presentation within the thesis.

Student signature: **Date: 22 July 2021**

The undersigned hereby certify that the above declaration correctly reflects the nature and extent of the student's and co-authors' contributions to this work. In instances where I am not the responsible author I have consulted with the responsible author to agree on the respective contributions of the authors.

Main Supervisor signature: **Date: 22 July 2021**

Acknowledgements

Doing a Ph.D. is like climbing a mountain—as we stood at the bottom of the mountain staring up at the summit, we recognized and felt all the fear, inadequacy, and doubt stirring inside us, but still chose to climb the mountain to achieve our goal—to bridge the gaps in a specific research area. Therefore, accepting right where we are, we started our journey, climbing our mountains. We learned as much as we could to make steps towards the summit. But, sometimes, we fell, backtracked, or got lost. We got tired. Our view got cloudy, and our legs got sore. The obstacles standing in the way of our success seemed to have exhausted our patience and confidence. At this very moment, the people who stood by us, those people we can trust, came to us to catch us before we fell. 'Keep climbing,' they said, 'for the path ahead but also for your climb so far.' Once again, we got back to work with a renewed gratitude for the moments we did feel the sunshine along our way. Finally, we reached the summit. We have accomplished a thing once upon a time we barely dared to dream. Looking down, we finally saw that all our obstacles were not setbacks but merely switchbacks getting us to our goal in the exact way we were meant to get there. We looked around and found that what we pursued is not the summit we are standing atop now, but the journey that witnessed our discovering, insisting, and becoming. We appreciate the people who stood beside us and assisted us with the correct path up the mountain. They provided the advice and encouragement necessary to get us over hurdles and gave us the energy and motivation to keep going forward.

My first and foremost thank goes to my supervisor Dr. Yuan-Fang Li who has supported me during the past two years and influenced me in many ways. He is an amiable man with wisdom and earnestness. His research insights always enlightened me with bright ideas, and his rigorous attitude and persistence to innovations stimulated my enthusiasm for scientific

research and inspired me to fight my way for my Ph.D. study. I thank him for his support of many insightful discussions that helped me understand my research problems and for his patience in going through all my drafts of my papers. He provided valuable suggestions for the problems that I encountered in my research and everyday life. I will benefit from what I have learned from him, not only for a Ph.D. degree but also for the rest of my life. He is not just a mentor to me—moreover, I regard him as a friend, as a noble person that I always trust, admire, and salute. I will never forget those days that we discussed the research ideas, wrote the papers together until late at night, and definitely, went hiking. I am deeply grateful to him.

I would also like to thank Dr. Gholamreza Haffari and Prof. Guilin Qi for steering me through my research journey and guiding me to become a better person. I greatly value the opportunity and experience to work with them. They are always there to respond to my questions or requests and consistently provide me with their best knowledge to enlighten me with ideas, suggest ways to hone my research skills and remind me of possible fallacies. I also deeply appreciate that they dedicated their time to read through my papers, comment on them, and help me edit them line by line so that they would be good enough to be accepted and published. I could not have done this work without these wise people being on my side.

I'd also like to thank the great collaborators, co-authors, and colleagues I've had the privilege of working with during my Ph.D. Thank you, Dr. Vishwajeet Kumar. Without your brilliant insights, we could not have published our paper. You are the only foreign friend I have ever made, and I will always remember the trips we went on, the games we played, the swimming we had, and surely, the Indian food you cooked for me. I would also appreciate all the people coauthoring with me, including Tongtong Wu, Yongrui Chen, Jingyao Zhang, Wei Wu, Daiqing Qi, and Yuhao Zhang. Your contributions further strengthened the technical quality and literary presentation of our paper.

I want to thank my roommates, lab-mates, my friends, and my colleagues at the Faculty of Information Technology during my study at Monash University, who are at different times and different organizations making my working days such a delight and having a good laugh together: Li Ji, Lin Wang, Lingling Shen, Guodong Ma, Ying Yang, Yifei Hu, Jishan Giti, Moataz Mahmoud, Maurice Ntahobari, Paula Bria, and Bhagya Gayathri Hettige. Thank Dr.

Bhagya, for your generous sharing of the materials relevant to writing a Ph.D. thesis. I'd also like to thank all the people with whom I have been working together at the lab in Southeast University: Weizhuo Li, Huan Gao, Tianxing Wu, Sheng Bi, Yiming Tan, and Nan Hu. Thank you for all those wonderful days we had together!

I would like to thank my panel members, Dr. Li Li, Dr. Lan Du, and Dr. Gholamreza Haffari, for serving on my thesis committee and providing many useful comments on the thesis. Their insights on my research work have helped me to extend my study and explore new research directions.

I would like to express my deepest gratitude to my family and close friends who check up on me periodically to see how I was doing in a country that is so distant from them. They consistently cared about my work and life when even I could not spend much time with them during this busy period. I always felt both warm and powerful whenever we connected, as they demonstrated their great faith in me and the importance of this work. This dissertation, and this incredible journey, could not have been accomplished without their constant love and support. Especially, I am grateful to my parents. They always stand by me, support me, and love me, regardless of my successes or failures. I appreciate my parents for bringing me up to become a better person and achieve my dreams. Last but not least, my deepest love is reserved for my wife, Renee Zheng, for her constant support, encouragement, and patience. This is as much her journey as mine, and I will spend the rest of my life grateful for her earnestness and endless love.

There were so many people in my life who impacted me that it would not be possible to list all of them here. I hope you can recognize yourself when reading these lines. Even so far away, you are so close to me.

Contents

List of Figures

Chapter 1

Introduction

Knowledge bases (KB) have quickly become an indispensable information source for research and practice in recent years. A great amount of effort has been invested into curating large KBs such as Freebase [1], DBPedia [2] and Wikidata [3]. Knowledge base question answering (KBQA) [4, 5, 6, 7, 8, 9], the task of interpreting natural-language questions as logical forms (such as SPARQL) which could be directly executed on a KB, has attracted substantial research interest as it is an accessible, natural way of retrieving information from KBs.

KBQA includes *simple questions* that retrieve answers from single-hop triples ("what is Donald Trump's nationality") [4, 6], *multi-hop questions* that infer answers over triple chains of at least 2 hops under specific constraints ("who is the president of the European Union 2012") [10, 11], and *complex questions* that involve set operations ("how many rivers flow through India and China") [12]. In particular, complex question answering (CQA) [13], the subject of this research, focuses on aggregation and multi-hop questions, is a sophisticated KBQA task in which a sequence of discrete *actions*—e.g., set intersection and union, counting, comparison—needs to be executed to derive the answer.

CQA is typically cast as a *semantic parsing* problem, whereby natural-language questions are transformed into appropriate structural queries (sequences of discrete actions). Such queries are then executed on the KB to compute the answer. Consider the complex question "How many rivers flow through India **and** China?" as a motivating example. Fig. 1.1 shows an incomplete sub-graph relevant to this question. To answer this question, all entities whose type

is "river" and link to the entity "China" with edge "flow" will first need to be retrieved from the KB to form the candidate set S_A. Meanwhile, the candidate set S_B will also be formed to represent those rivers that flow through India. After obtaining the intersection of S_A and S_B, the number of elements in the intersection can finally be identified as the correct answer to the question. It can be seen that a diverse set of operations, including selection, intersection, and counting operations, need to be sequentially predicted and executed on the KB.

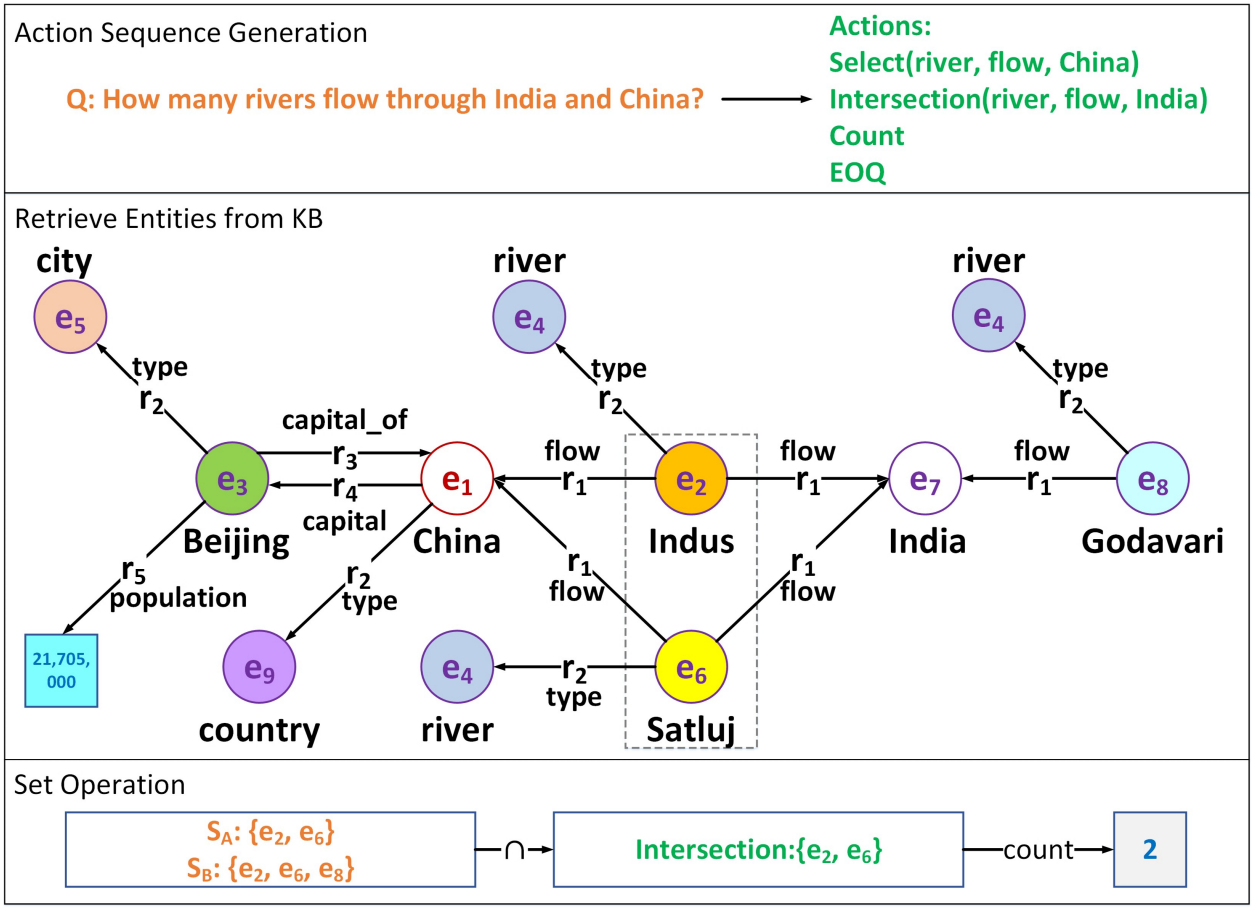

Figure 1.1: A running example illustrating the task of complex question answering and the relevant KB snippet. The first row shows the transformation of a natural language question into the corresponding logic form. The second row illustrates the snippet of the underlying KB, which is relevant to the query. The third row indicates the set operations required to answer the question.

Sequence-to-sequence (seq2seq) models learn to map natural language utterances to executable programs and are thus good model choices for the CQA task. However, under the supervised training setup, such models require substantial amounts of annotations, i.e., manually annotated programs, to train effectively. For practical KBQA applications, gold annotated programs are expensive to obtain, and thus most of the complex questions are not paired with the annotations [12]. Reinforcement learning (RL) is an effective method for training KBQA

models [11, 14] as it does not require annotations but only denotations (i.e., answers) as weak supervision signals. By marrying two powerful ideas—representing natural language utterances with continuously distributed vectors and executing symbolic programs for reasoning—Neural Program Induction (NPI) models are proposed to solve the CQA problem under the RL paradigm. However, NPI-based KBQA methods face several significant challenges.

Sparse reward. Neural-symbolic models, the models that take the NPI approach, have been proposed for the CQA task [11, 12, 14]. In the RL context, questions of the same pattern could be regarded as one single *task*, while programs trying to solve these similar questions are considered *trials*. Instead of using the gold annotations, neural-symbolic models employ *rewards*, i.e., comparisons between the predicted answer and the ground-truth answer as the distant supervision signal to train the policy [11, 14]. Usually, a positive reward could only be given at the end of a long sequence of correct actions. However, in the initial stage of model training, most of the trials sampled from the sub-optimal policy attain small or zero rewards [15]. Thus, this *sparse reward* problem in the CQA task is a major challenge that current neural-symbolic models face.

Data inefficiency. Furthermore, due to the sparse supervision signals, such models are often *data inefficient*, which means many trials are required to solve a particular task [16]. Being trained from scratch, RL models often need thousands of trials to learn a simple task, no matter what policy is employed to search programs.

When faced with a large number of questions, training such models would consume an enormous amount of time. One way to increase data efficiency is to acquire task-related prior knowledge to constrain the search space. However, in most cases, such prior knowledge is unavailable unless manual labeling is employed. Hence, the data inefficiency problem often makes models expensive to train and thus infeasible/impractical. Liang et al. [11] proposed the Neural Symbolic Machine (NSM) that maintains and replays *one* pseudo-gold trial that yields the highest reward for each training sample. When using RL to optimize the policy, NSM assigns a deterministic probability to the best trial found so far to improve the training data efficiency. However, in NSM, the best trial to be replayed might be a spurious program, i.e., an incorrect program that happens to output the correct answer. Under such circumstances, NSM

would be misguided by the spurious programs since such programs could not be generalized to other questions of the same pattern. Besides, NSM only harnesses the accuracy of the predicted answers to measure the reward, hence also suffers from the sparse reward problem.

One-size-fits-all. The conventional approach to CQA is to train *one* model to fit the entire training set and then use it for answering all complex questions at the test time. However, such a one-size-fits-all strategy is sub-optimal as the test questions may have diversity due to their inherently different characteristics [17]. For instance, in the CQA dataset, the samples could be categorized into seven different types, e.g., those capturing logical/comparative/quantitative reasoning. The length and complexity of questions in one group are likely to differ from those in other groups. Therefore, action sequences relevant to different groups may have significant deviations. It is hard to learn a one-size-fits-all model that could adapt to varied types of questions. An exception is [18], where it proposes a few-shot learning approach, i.e., S2A, to solve the CQA problem with a retriever and a meta-learner. The retriever selects similar instances from the training dataset to form *tasks*, and the meta-learner is trained on these tasks to learn how to quickly adapt to a new task created by the target question of interest at the test time. However, Guo et al. [18] use teacher forcing within the learning by demonstration approach, which suffers from the drawback that the demonstrations are often hard to collect. Also, the retriever is trained independently of the meta-learner. Thus, the result of the meta-learner answering the questions is irrelevant to the retriever. Therefore it is hard to evaluate the quality of the support set that the retriever establishes for each new question and consequently tricky to measure the retriever's impact on answering the questions. If the samples found by the retriever are not similar to the new question as expected, the meta-learner will be misguided by the deviated examples and thus learns a model that is not suitable for the current task.

In the following section, we present the research questions (RQ) to define our research study to develop a neural-symbolic framework to solve the CQA task while addressing the challenges mentioned in the literature review.

1.1 Research Questions

In this research study, we formulate the following RQs with the primary focus of effectively learning a CQA model that could precisely answer the complex questions.

RQ1: How to design a CQA framework by only using denotations while alleviating the sparse reward and data inefficiency problems that inherently lie in the conventional NPI models?

To mitigate the problem, we present an RL architecture that incorporates curriculum learning and memory-buffer.

RQ2: How to learn an adaptive model to precisely answer each novel question instead of using a fixed global optimal model, aka one-size-fits-all model, to solve the CQA task?

To solve the problem, we propose a meta reinforcement learning (meta-RL), where the model adapts to the target question by *trials* and the corresponding reward signals on the retrieved instances that form the task.

RQ3: How to design an optimal retriever to find the most appropriate instances for the target question?

We propose a new learning algorithm that jointly optimizes the retriever and the adaptive CQA model to address the problem.

1.2 Motivation

This section provides a detailed introduction to each research question's motivation and how we addressed each question.

1.2.1 RQ1: How to design a CQA framework by only using denotations while alleviate the sparse reward and data inefficiency problems that inherently lie in the conventional NPI models?

Due to the difficulty of collecting annotations, the existing CQA dataset [13] only contains the denotations for each question. The literature takes one approach to deal with the missing annotations, which aims to transform learning a model for CQA into learning by demonstration, aka imitation learning, where a *pseudo-gold* action sequence is produced for the questions in the training set [19]. This is done by employing a blind search algorithm, i.e., breadth-first search (BFS), to find a sequence of actions whose execution would yield the correct answer. This pseudo-gold annotation is then used to train the programmer using teacher forcing, aka behaviour cloning. However, BFS inevitably produces a single annotation and is ignorant to many other plausible annotations yielding the correct answer.

To alleviate this issue, following the previous literature [11, 20], we propose a framework Neural-Symbolic Complex Question Answering (NS-CQA) that is based on RL to make use of the search policy prescribed by the programmer. Not only this addresses the limitation on the 1-to-1 mapping between the questions and annotations, but also it enjoys a better computational footprint compared to BFS as the RL policy amounts to an informed search.

Moreover, instead of recording one trial for each question, we maintain a memory buffer to store the promising trials, i.e., the action sequences that lead to the correct answer or gain high rewards. On the one hand, in our work, we aim to converge a behavior policy to a target optimal policy. Thus we need to measure how similar/important the generated trials are to trials that the target policy may have made. We design a reward bonus, **proximity**, to favor these "similar/important" trials. On the other hand, to avoid overfitting the spurious trials, we also encourage the policy to explore in undiscovered search space, i.e., the space beyond the imitation of the pseudo-gold annotations. Therefore, we design another reward bonus, **novelty**, to the generated action sequences that are "different" from the trials stored in the memory buffer.

To adaptively control the exploration-exploitation trade-off, we employ a *curriculum learning* [21] scheme to dynamically change the influence of the two reward bonuses, namely the proximity and the novelty. Mainly, given a question, we define the proximity for the predicted trial as the highest similarity between the predicted trial and all the trials in the memory buffer. Novelty is also defined through similarity. We consider a trial is novel if it is not similar to all the trials in the memory buffer. At the initial stage, since the policy is sub-optimal and the trials generated by the policy are generally infeasible, we prefer more novelty for exploration. We gradually increase the proportion of proximity and reduce the influence of novelty during the later training epochs. At the final stage of the training, the proportion of proximity in the reward bonus will rise to 100%. Such a delicately designed curriculum method can significantly improve the learning quality and efficiency [22].

Besides, to alleviate the sparse reward problem, an adaptive reward function (ARF) is proposed. ARF encourages the model with partial reward and adapts the reward computing to different question types. We sum up the reward bonus with the adaptive reward to make our RL model learn from the combined reward. With this modification, we reshape the sparse rewards into dense rewards and enable any failed experience to have a nonnegative reward, thus alleviate the sparse reward problem.

Furthermore, by incorporating a memory buffer, which maintains off-policy samples, into the policy gradient framework, we **improve the sample efficiency** of the REINFORCE algorithm in our work [16]. We encourage the model to generate trials similar to the trials in the memory buffer using the proximity bonus. Therefore the high-reward trials could be re-sampled frequently to avoid being forgotten in the training process. Since a group of question shares the same pattern, a sequence of the same actions could solve such questions. Once we improve sample efficiency, we reduce the trials needed for training. Therefore, using a minimal subset of training samples, our model can produce competitive results.

To sum up, we proposed a novel RL-based CQA framework in our research study, proving to outperform the existing state-of-the-art CQA models.

1.2.2 RQ2: How to learn an adaptive model to precisely answer each novel question instead of using a fixed global optimal model, aka one-size-fits-all model, to solve the CQA task?

Since a one-size-fits-all strategy is sub-optimal as the test questions may have diversity due to their inherently different characteristics [17], we propose a framework that could adaptively form a set of parameters, which is optimal for the current new question, via online learning. We present a meta-RL approach for CQA (MRL-CQA), where the model adapts to the target question by *trials* and the corresponding reward signals on the retrieved instances. In the meta-learning stage, our approach learns an RL policy across the tasks for both (i) collecting trial trajectories for effective learning and (ii) learning to adapt programmer by effectively combining the collected trajectories.

The purpose of the meta-RL approach is to change the training goal of the one-size-fits-all NPI model, that is, to find an optimal initialization parameter instead of finding a global optimal parameter to fit all training samples. By doing so, the model can adaptively generate unique parameters for each task based on the optimal initialization parameters, thus accurately answering the questions, improving the model effect, and quickly learning and adapting to new tasks. Thus, the tasks generated from a tiny (less than 1%) portion of the training data are sufficient for training the meta learner.

As a deliverable in this research component, we improve the RL-based CQA framework with meta-learning.

1.2.3 RQ3: How to design an optimal retriever to find the most appropriate instances for the target question?

To automatically learn the optimal retriever and find the most similar instances for meta-learning, we propose MetA Retrieval Learning (MARL), a new learning algorithm that jointly optimizes retriever and programmer in two stages.

In the first stage, we fix the retriever's parameter and employ it to select the top-N similar questions (secondary questions) to a given target question (primary question). The *trial* trajectories, along with the corresponding rewards for answering each secondary question, are used to adapt the programmer to the current primary question. The feedback on how correctly the adaptive programmer answers the primary question is used to update the programmer's weights.

In the second stage, we fix the programmer's parameter and optimize the retriever. The *general* programmer first outputs an answer and gain a reward to the primary question without using any secondary questions. Then we sample M different sets of secondary questions following the retriever policy and employ the question sets to learn M different programmers. Each specific programmer will generate an answer to the primary question and gain some reward. We consider the difference between the reward yielded by the general programmer and the reward gained by each adapted programmer as the contribution of employing the corresponding support set. Thus, the reward difference provides the training signal to optimize the retriever's policy: a positive difference increases the probability that a set of secondary questions is chosen, and a negative difference lowers the probability.

We train the programmer and the retriever alternatively until the models converge. Note that in our method, the training of the retriever is done in a weakly-supervised manner. The retriever is optimized to find better support sets according to the programmer's performance of answering primary questions, rather than employing teacher forcing. Since one support set generates one adaptive programmer, we employ the feedback obtained by evaluating the adaptive programmer as a weak supervision signal to optimize the retriever. Therefore we encourage the retriever to find the support set leading to a superior programmer that gains more reward. At the same time, the programmer is optimized alongside the retriever.

Therefore, we design a meta-learning based approach that could optimize the retriever and meta-learner alternately to learn a KBQA system applicable to questions with diversity.

1.3 Thesis Outline

The rest of this document is sectioned into five chapters. Chapter 2 is devoted to a summary of the related previous literature in the area of research. In the following chapters 3, 4, and 5, our research contributions are presented in the published research articles. Chapter 6 concludes the thesis with remarks on our research work and discusses the potential future research directions. A brief overview of each chapter is provided as follows.

Chapter 2: In Chapter 2, we review the literature relevant to our research study. This chapter builds a solid foundation for our research study by formally defining the terminologies and concepts related to the KBQA task. This chapter also explores several branches of works that adopt different algorithms and frameworks to solve the KBQA task. More importantly, we analyze the challenges encountered in the approaches that focus on the KBQA problem.

Chapter 3: In this chapter, we present our proposed RL-based model, NS-CQA, which models the CQA task as a neural-symbolic learning task. NS-CQA is trained to learn a policy that could produce programs mapping to the input natural language questions (NLQ) with distant supervisions, i.e., the program execution results. Meanwhile, NS-CQA attempts to alleviate the challenges in the previous NPI models, including sparse reward and data inefficiency problems.

Chapter 4: This chapter discusses the contribution of our research study in solving the one-size-fits-all problem. When complex questions vary widely, the state-of-the-art methods are often hard to find a set of optimal parameters for all questions. Instead, we aim to find a set of the initial parameters by which multiple sets of parameters could be adaptively generated for answering multiple questions, respectively. We achieve this target by integrating our neural-symbolic model with a meta-learning method, proposing MRL-CQA.

Chapter 5: MARL is an extension of the previous research work, MRL-CQA, which optimizes the retriever along with the meta learner. In this work, we verify that the quality of the support

set used for meta-learning influences the whole system's performance. Therefore, we design a method to train the retriever to produce better support sets under the weak-supervised learning paradigm, dispensing with the need for retriever annotations. Moreover, we manage to unify the retriever and the meta learner optimization into a single learning process.

Chapter 6: The final chapter summarises the research conducted so far by highlighting the main contributions of each research component. Also, at the current stage of the research, we identify limitations in our research study and discuss several exciting future research directions branching from our research work.

Chapter 2

Literature Review

This section introduces the concepts necessary for understanding the field of Knowledge Base Question Answering (KBQA) in detail, including a formal definition of Knowledge Base (KB), a definition of KBQA tasks, a description of logical form, and an introduction of the several lines of approaches applied to KBQA tasks. At the end of this chapter, we summarize what we have achieved in designing our KBQA model.

2.1 KB and KBQA

Web search has been going through a significant change—from simple document retrieval to natural language question answering [23]. Instead of searching for documents that match requested keywords, users seek a solution that could directly return precise answers to their questions. Therefore, a search needs to understand the intention of users' NLQs fully and selects relevant facts to infer the answers [24]. To achieve this, researchers make an effort to design a data infrastructure, i.e., KB (which is also called knowledge graph, i.e., KG[1]), to store and manage the ever-growing knowledge in a structured format, and thus ease question answering [25]. Therefore, compared with document retrieval, KBs offer a structured format more readable for the machine and provide a more natural way of interacting with knowledge regarding a particular domain or a general domain [26].

[1]The phrases *knowledge graph* or *knowledge base* are often used interchangeably in this thesis, following the previous literature.

KB consists of millions of *entities* denoting the subjects of interest in a specific domain, and thousands of *relations*[2] denoting the interactions between these entities. KB could also be viewed as a way of structuring facts in a multi-relational graph form where *entities* are viewed as *nodes* and *relations* between these entities are represented as *edges*. In KB, each fact is represented in triple format, i.e., (*head entity*, *relation*, *tail entity*)[3], indicating that two entities are connected by a specific labeled edge, aka a relation [27]. For example, Beijing and China can be entities corresponding to the real-world city of Beijing, and the country China respectively. Moreover, capital_of is a relation between the above two entities denoting that Beijing is the capital of China. Therefore, the entities and the relation compose a triple (Beijing, capital_of, China). Other than *entities*, a *relation* can also link an entity with an attribute value (i.e. a date, a number, a string, etc.), denoting as a *string literal* [28]. For instance, the entity Beijing could be connected via a relation population to a integer value 21,705,000 indicating that the current population of Beijing is 21,705,000 in real world. Figure 1.1 depicts a snippet of an example KB representing the facts relevant to the above instances.

In this research study, $\mathcal{E} = \{e_1, ..., e_{n_e}\}$ denotes the set of *entities*, $\mathcal{L}$ is the set of *string literals*, and $\mathcal{R} = \{r_1, ..., r_{n_r}\}$ represents the set of *relations* connecting two entities or linking an entity with a literal value. We denote the set of all the possible ordered combination of the elements of $\mathcal{E}$, $\mathcal{L}$, and $\mathcal{R}$ as $\mathcal{E} \times \mathcal{R} \times (\mathcal{E} \cup \mathcal{L})$, where the first position in a combination could be entities of $\mathcal{E}$, the second position could be instantiated with relations in $\mathcal{R}$, and the third position could be elements of $\mathcal{E}$ or $\mathcal{L}$. Among all these random combinations, some of them are feasible and meaningful. We thus define a triple $t \in \mathcal{E} \times \mathcal{R} \times (\mathcal{E} \cup \mathcal{L})$ to describe a fact involving a head entity, a relation, and a tail entity or a string literal, representing a piece of knowledge in the real-world settings, and therefore describe a KB $\mathcal{K}$ as a subset of $\mathcal{E} \times \mathcal{R} \times (\mathcal{E} \cup \mathcal{L})$.

KBs integrate heterogeneous data and information on the Web into a consistent, unified formal knowledge representation, making the complicated information easier to understand and represented by machines. Therefore, KBs organize the knowledge on the Web in a unified way and express the massive information on the Internet in a form that is more in line with human cognition and understanding, making the relevant knowledge more closely connected, and facilitating people to obtain and trace the required knowledge [29]. In recent years, as the foundation of knowledge interconnection,

[2]The words *predicate* or *property* are often used interchangeably with *relations* in this thesis, following the previous literature.

[3]In some work, the *head* and *tail* entity are represented as *subject* and *object* entity respectively.

many high-quality knowledge bases covering a large amount of knowledge have been constructed, which provides strong support for knowledge search and has become the infrastructure of Internet cognitive intelligence services. Since Wikipedia[4] is the largest shared knowledge resource database established by the collective intelligence, many researchers have built huge knowledge bases based on Wikipedia knowledge, such as DBPedia [2, 30]—extracting knowledge items from Wikipedia to build an ontology, YAGO [31]—combining the conceptual topology of word network in WordNet [32] with the entities in Wikipedia, Freebase [1]—the structured KB constructed based on Wikipedia, and Wikidata [3]—the KB treating the extracted knowledge on the Wikipedia pages as the information source. Besides, a great amount of effort has been invested into curating large KBs with information source other than Wikipedia, such as NELL [33], BabelNet [34], Microsoft Concept Graph [35], and ConceptNet [36]. In practice, the KBs have been successfully applied to many real-world applications, including semantic parsing [4, 37], named entity disambiguation [38, 39], information extraction [40, 41], and question answering [25, 42].

Google formally proposed KB on May 17th, 2012 [43] as the semantic enhancement of Google's search engine to enable searching for more than literal strings, which are real-world objects. In other words, Google's original intention of proposing KB is to improve the quality of search engines, provide users with fine-grained knowledge and meet the increasing information needs of users. The traditional searching method is that the search engine uses keyword matching technology to return a list of candidate coarse-grained web pages containing keywords in the users' questions. Then the users select the most relevant web pages from these candidates and recognize the information they need. The semantic searching functions, combined with KB (such as Google's search engine), have certain "associative ability", which could provide relatively fine-grained answers. Such searching functions return the entities and concepts associated with search terms to users in the form of "knowledge card" [44] according to the KB's structure and the query's intention. As shown in figure 2.1, when a user enters the keyword "Shanghai" in Google to search, the search engine will return a knowledge card containing the relevant information of "Shanghai", instead of just returning the web page containing the search keyword "Shanghai". However, the knowledge card returns to the user is still about the information searched based on keyword matching and does not possess the ability to "understand" the problem. Moreover, the KB often contains many triples, and the topological relationships among the triples are complex. Due to the complex structure, it poses a great challenge for the common uses to access the huge KB.

[4]https://en.wikipedia.org/wiki/Main_Page

Figure 2.1: An example of a knowledge card about Shanghai, returned from the Google engine.

Large KBs constitute the underlying data storage foundation and information source of various applications. Therefore, the ability to query the information stored in the knowledge base through the natural language interface is of practical importance [45]. To lower the threshold for users to search KBs and return the relevant information inquired by users accurately, KBQA—an advanced information retrieval method that bridges the interface between human users and machine intelligence for information exchange, has been proposed by researchers [46]. The purpose of the QA system is to understand the questions put forward by users and to transform natural language questions into standard structured queries. By executing structured queries on KBs, information such as related entities and attribute values in KBs can be obtained, which can be returned to users as answers to questions to facilitate users to obtain desired knowledge. QA systems hide the details of question comprehension, knowledge retrieval, and logical reasoning and thus provide a simple interface to the users. Therefore, in practice, users only need to pose NLQs to get fine-grained and accurate answers.

Based on the rapid development of artificial intelligence, KBQA is a new technology that can quickly acquire the knowledge that users need, thus attracting wide attention in academic and industrial circles and becoming a hot issue in recent years [4, 5, 7]. At present, KBQA has become one of the important driving forces to promote human-computer interaction and Internet semantic search, and it also plays a powerful role in many practical, intelligent applications. First of all, KBQA can provide users with a more natural way than keyword search to interact, especially for those users who are not familiar with information technology [47]. Therefore, KBQA provides an interactive interface for information systems that can greatly improve usability [48]. Secondly, the KBQA system can effectively accelerate the search process in that users can directly get the correct answers to questions [49]. In fact, this avoids a lot of manual work needed to identify related documents and locate correct information in a document. One of the most well-known KBQA systems is IBM Watson [50], which is famous for winning the game show "Jeopardy" in 2011. Since then, IBM Watson, as an effective QA technology, has become the basic component of business intelligence [51]. In real-world scenarios, the QA functions are also used to provide medical care decision support based on clinical literature [52]. In addition, in other fields, the QA systems aiming at practical application have been proposed. For example, KBQA has been used in education [53] and information technology security fields [49]. In these fields, a QA system with corresponding professional knowledge can provide better services to users. Besides, many dialogue systems and QA systems have introduced KBQA to meet users' complex information needs and improve the QA experience. The typical applications include IBM Watson system, Microsoft Cortana, Baidu's Xiaodu robot, Google Search, Bing search, Google Assistant, Siri Voice Assistant, AliMe, and other dialogue assistants [54, 55].

To sum up, on the one hand, KBQA has become an interface for users to retrieve KB knowledge; on the other hand, it has practical application value in many human-machine dialogue/question answering systems. However, due to the complex characteristics of the NLQs, it is challenging to solve the KBQA task, especially the CQA task. How to design an effective QA method for complex questions, understand the intention of the NLQs raised by users, and accurately retrieve the fine-grained knowledge inquired by users from the underlying KB is a subject with important practical significance and research value.

2.2 Tasks in KBQA

Although useful in organizing data, KBs are tough for users to access due to the complex underlying structure. Several structured query languages [4, 56, 57] and custom-defined actions [11, 19, 58] have been proposed for accessing the structured data in KBs. However, using them requires a certain familiarity with the facts maintained in KBs as well as an in-depth understanding of the syntax and semantics of the logical forms [24, 59, 60]. Neither of these an average user can be assumed to possess. To expand the applicability across broader users, a new research direction known as KBQA [4, 5, 6, 7, 8] is proposed. KBQA is users raising NLQs in their own terminology and receiving concise answers from KBs. Using KBs as a knowledge source, the approaches for KBQA usually model the KBQA task as an interpretation of natural-language questions as logical forms that could be directly executed on KBs [61]. Therefore, determining how to translate NLQs to logical forms automatically is the core goal of the KBQA task, which has attracted substantial research attention lately [62, 63, 64, 65].

2.2.1 Definition of KBQA

The approaches to KBQA, especially the state-of-the-art ones, employ a *semantic parsing framework* to accomplish the KBQA tasks [66, 67, 68, 69, 70]. The semantic parsing framework aims to translate a NLQ into logical forms that could be executed on KB to get the answer [61, 71, 72]. In this research study, the logical form is denoted as *annotation*. In contrast, the logical form's execution result or the answer for the question is referred to *denotation* [73].

Therefore, we define the KBQA task as follows. We denote KB as $\mathcal{K}$, a NLQ as q, and the set of all possible answers as $\mathcal{A}$. $\mathcal{A}$ involves the union of three sets, each refers to one answer type: (i) the set of *entities* $\mathcal{E}$ and *string literals* $\mathcal{L}$ in $\mathcal{K}$ (for example the answers to "capital of China" and "population of Beijing"), (ii) the numerical results of the aggregation functions including COUNT or SUM (answer to "how many rivers flow through China"), and (iii) the Boolean value set {True, False} that is for true/false questions (answer to "is Beijing in China"). We thus define the task of KBQA as to yield the correct answer $ans \in \mathcal{A}$ corresponding to the given question q.

To answer the given question q correctly, in this research study, we aim to conduct *semantic parsing*, that is, mapping q into a logical form τ. Each logical form τ consists of a sequence of actions

$\{a_1, ..., a_t\}$ ($\tau = \{a_1, ..., a_t\}$), where the action token a_i could be: (i) arithmetic/logic aggregation functions available in the query languages or custom-defined functions (refer to 2.3.2), (ii) KB artifacts that used to insert in the functions, and (iii) the numeric tokens appear in the question (for instance "10" in the question "what countries have more rivers than 10"). Let T be the set of logical forms constructed by combining all possible action tokens. The semantic parsing task aims to select the logical form τ from T ($\tau \in$ T) that could lead to the ground-truth answer ans_g and fully express the meaning of the question q. The latter constraint is important because some logical forms might coincidentally output the correct answer ans_g but could not cover the full meaning of the query q. We refer to these incorrect logical forms as *spurious* logical forms [73, 74]. For instance, when facing the utterance "Which musical ensembles were formed at Belfast?", the expected logical form should be "Select(Belfast, location, musical ensemble)". However, the logical form "Select(Belfast, location, musical ensemble), Inter(Belfast, location, musical ensemble)" could also yield the desired answer. If we analyze the second logical form, we could find that the second action, 'Inter' is unnecessary to this question. We view this logical form as a typical *spurious* problem since it cannot represent the original meaning of the question. Therefore, we aim to design a semantic parsing framework by which the correct logical form τ that yields the gold answer ans_g and fully conveys the meaning of question q could be successfully selected from the set T.

2.2.2 Subtasks in KBQA

Other than the performance of the semantic parser, the accuracy of the relevant QA subtasks including *entity linking* and *predicate recognition* also affect the KBQA systems [12]. The goal of the subtasks is to analyze the question to clarify what objects the question refers to, and identify the relations between these objects. To accomplish this goal, the KBQA systems often resort to relevant methods to detect the entity, type, and predicate mentions in the questions, and map them into their counterparts in KB [75]. KBQA systems always implement independent components to solve the above subtasks respectively and arrange them into a KBQA pipeline [76, 77, 78].

Entity linking is the task of identifying which entity in KG is referred to the entity mention, i.e., a certain phrase or a span of words in question q. The task of entity linking is normally divided into two steps. Take the question "what is the country that Indus River flows through" as an example question, the first step is to detect the entity mention in q, as to detect "Indus River" in the example. Entity mention detection could be regarded as a classification problem in which a word would be classified

into positive class if it is a part of an entity span, or negative class indicating the word dose not belong to of an entity mention [79]. Or it can be viewed as a sequence tagging problem where all tokens in the question will be tagged with sequential labels to indicate whether a token belongs to the entity mention or not [80]. Even assuming that the entity mentions in the question have been accurately detected, considering the size of KB, the ambiguity of entity names still make it a challenge to pick the correct entity from a large number of candidates. The larger the size of the KB is, the higher the possibility that different entities will share a same name [81]. Then, the second step of entity linking is to ground the entity spans to their corresponding entities in a KB. In the above example, we need to link the mention "Indus River" to the entity "Indus" in the KB. When comparing the surface forms are not enough to disambiguate the entity mention to the target entity, it is essential to consider the contextual tokens surrounding the entity mentions. Also, as the size of KB keeps increasing, the number of the entities becomes too large to establish the lexical mappings. Therefore the statistical entity linking approaches are required to generalize to entities that are unseen in the training corpus. A series of KBQA systems introduce methods that are specifically targeted at solving the entity linking task. For instance, DBpedia Spotlight [82] is devoted to mine DBpedia's entities and S-Mart [83] is proposed for Freebase. Zhu et al. [84] apply the Wikipedia Miner tool[5] [85] to identify the entity mentions in the questions and return the entity linking results over Wikipedia.

Predicate recognition aims to determine the predicates that are used to compose the logical forms. By mapping the certain phrases in the natural utterance to the corresponding KB's predicates, predicate recognition task bridges the gap between the natural expression and KB vocabulary. In our example question "What is the capital city of the country that Indus River flows through?", the predicate "capital" is referred to the *pattern* "what is the capital city of", which consists of noun and verb phrases. However, it remains nontrivial to conduct this task since a predicate could refer to various natural language expressions, while these expressions are often very different from the surface forms of the predicate [81]. For instance, the relation "population" can be expressed as "what is the population of", "how many people live in", "what is the number of people in", etc. It can be seen that in some cases, even though the surface form of the predicate "population" is not explicitly mentioned in the question, the corresponding predicate still needs to be correctly inferred. This makes demands on the generalization ability of the KBQA system when solving the predicate recognition task. Therefore, attempts have been made to employ specialized models to solve the task in several KBQA systems. For instance, Berant et al. [4] make a coarse mapping between the predicates and

[5]http://wikipedia-miner.cms.waikato.ac.nz/

natural language phrases, then use a bridging operation to generate additional predicates based on the adjacent predicates. Yih et al. [8] apply a convolutional network to match relations and questions. Yu et al. [9] propose a KBQA system that uses deep residual bidirectional LSTMs to match questions to relation names with word-level and character-level.

Several modern systems attempt to develop a single model to solve entity linking and predicate recognition tasks jointly. To answer complex questions over a KB, Ansari et al. [14] propose a query annotator to conduct joint entity, relation, and type linking in an unsupervised setting. Knowledge Embedding based Question Answering (KEQA) [81] proposes a KG embedding framework to jointly infer the head entity and predicate. KEQA projects the NLQ into entity and predicate embedding representations respectively, then retrieves the fact in the KG whose embedding is the closet to the entity and predicate embedding. Also, in broader natural language processing (NLP) spectrum, there also exist methods to solve the two tasks concurrently. Entity and Relation Linker (EARL) [86] models entity linking and relation linking as a Generalised Travelling Salesman Problem (GTSP) task, and uses GTSP approximate algorithm to solve the task. Yuan et al. [87] develop Relation-specific attention network (RSAN) to extract entity and relation jointly. RSAN utilizes attention mechanism to form specific representation of the question for each relation and apply sequence labeling to identify entities.

2.3 Logical Form

The task of answering a complex NLQ is to learn to map the question into a *logical form* that could be executed directly on a structured knowledge base to compute the answer [73]. The core part of the QA task is to design a semantic parser to translate the questions into logical forms [88]. The logical form could be well-defined query languages, including Prolog [56, 89], SQL [90], graph queries [91], FunQL [92], SPARQL [93], Lambda Calculus (λ-calculus) [94], Dependency-Based Compositional Semantics (DCS) [66], and Lambda Dependency-Based Compositional Semantics (λ-DCS) [95], or other custom-defined actions[6] which are built upon the query languages to simplify query form, reduce redundancy, and supply generalization ability.

[6]The words *action*, *action sequence*, or *sequence of actions*, are often used interchangeably with *program* in this thesis, following the previous literature.

2.3.1 Query Languages

The query languages are formally defined in a context-free grammar (CFG), and can be used for complex fact retrieval involving logical operations (i.e., disjunction, conjunction, and negation), aggregation functions (including grouping, counting, and comparing), filtering under certain conditions, as well as other ranking mechanisms [68]. Let's have a brief look at the following representative query languages.

SPARQL, a recursive acronym for SPARQL Protocol and RDF Query Language[7], is a standard query language to search and explore data stored in Resource Description Framework (RDF)—that is, a general data model that uses triples, where each is made of 3 resources of the form (*subject*, *predicate*, *object*), to represent data and compose a KG [96]. When retrieving triples on such a graph, the basic block for constructing relevant SPARQL queries is called *basic graph pattern* (BGP) [97]. A BGP consists of a set of RDF triples in which the query *variables* might lie at the subject, predicate, or object positions. By integrating with filters, optionals, expressions, aggregations, unions, differences, or ordering, BGP can expand into more complex graph patterns and form a SPARQL query. The result of the SPARQL query is defined as the multiple KG resources (i.e., entities, relations, or literals) that could map to the variables in the SPARQL when conducting graph pattern matching—that is, matching the graph pattern of the SPARQL query against the KG [98]. The example below shows a SPARQL query to answer the NLQ "What is the capital city of the country that Indus River flows through?" over the example KB in figure 1.1. The corresponding SPARQL query is:

```
SELECT DISTINCT ?uri
WHERE
{
    dbr:Indus dbo:flow ?x .
    ?x dbo:capital ?uri .
}
```

The query consists of two parts: the SELECT clause indicates the *variables* to list after executing the query, and the WHERE clause describes which triples to pull from the KB when querying[8]. The WHERE clause does this with one or more triple patterns, which like triples but some positions are

[7]https://www.w3.org/2009/sparql/wiki/Main_Page
[8]https://www.w3.org/TR/sparql11-query/

substituted with *variables* as wildcards. Like the above example shows, in the WHERE clause, the object position in the first triple pattern is replaced with the variable ?x. Therefore, this triple pattern will match against triples in the example KB whose predicate is the flow property, whose subject is Indus and whose object is anything at all, since this triple pattern has a variable that is named ?x. The query processor therefore assigns the object value to the ?x variable, looks for the triples that match the second triple pattern, and views the objects in the retrieved triples as the answers. By executing the SPARQL query against the example KB in figure 1.1, the result is Beijing.

λ-calculus is a formal function abstraction invented based on Church's Thesis [99]. This formalism defines the notion of computable function and has provided a solid theoretical basis for the functional programming languages [100]. λ-calculus acts as a well-defined, machine-interpretable meaning representation in some semantic parsing tasks [101]. More specifically, in the KBQA task, λ-calculus is introduced as a formal query language expression [8, 61]. The central part in λ-calculus is that of "*expression*", which applies *function* to the *variable argument* that is marked with the Greek letter λ, and get a value [102]. The example of the question "What is the capital city of the country that Indus River flows through?" could be mapped into the following λ-calculus expression: $\lambda x.capital(\lambda y.flow(Indus, y), x)$[9]. This expression contains the following parts: *constants*, which can be *entities* (Indus) or *functions* (flow); *variables*, the placeholders for the entities or numbers (x and y); *lambda expressions*, which represents functions acting on variables or constants (for example $\lambda y.flow(Indus, y)$ stands for the objects that Indus River flows through). Though SPARQL and λ-calculus have well-established grammars to express NLQs, the key of such expression is to understand the meaning of the question. Most existing methods typically rely on compositional grammar, such as compositional combination grammar (CCG) [103], to conduct question understanding. To train the parser, such methods require manually annotated combination grammars, which are relatively expensive to collect. Moreover, this mechanism assumes a fixed and predefined lexicon, limiting its capability of scalability when answering questions over a large-scale KB [24]. To overcome this challenge, another formal language called DCS is proposed in [66]. It provides a tree structure named DCS tree to construct logical forms, making parsing and learning more convenient. Unlike DCS, the formal language λ-DCS [95] is not restricted to the tree-structure, making it more expressive. Also, by removing the use of variables and making quantifiers implicit, λ-DCS could provide more compact logical forms than λ-calculus. λ-DCS has been applied to KBQA [4].

[9]https://plato.stanford.edu/entries/lambda-calculus/

2.3.2 Custom-defined Actions

Although query languages have been provided for accessing KBs, only experienced researchers who have been familiar with the grammar know how to utilize them [24]. The complex grammar and structure of the query languages make it an obstacle for ordinary users to use. Furthermore, most of these logical query languages are implemented in a subset of first-order logic [104]. However, the verbosity and the syntactic limitation make first-order logic hard to use [105]. Such practical drawback of the query languages (such as SPARQL and λ-calculus) usually leads to structurally and syntactically overcomplex expressions [68]. Some researchers built domain-specific actions based on query Languages to simplify the query format, reduce verbosity, and adapt to different QA tasks to tackle these challenges.

Neural Symbolic Machines (NSM) [11] proposes and implements a set of *functions* in high level programming language for semantic parsing. The generalizability, compositionality and scalability of the functions makes them adapt to varying input sizes more easily. The functions take several arguments (the argument could be entity or relation in KB, or a variable) as input, return a list of entities as execution result, and assign the entity list to a new variable. By employing such custom-defined functions, NSM could compose the programs necessary for semantic parsing, which equals to a subset of λ-calculus [8]. For example, the mathematical definition of a function is as follows: $(Hop\ r\ p) \Rightarrow \{e_2 | e_1 \in r, (e_1, p, e_2) \in \mathcal{K}\}$. The object entities in the triples whose subject is one of the entities in set r and whose predicate is p are obtained as the result of executing this function. The result is saved in a new variable for further use.

Complex Imperative Program Induction from Terminal Rewards (CIPITR) [58] presents twenty *operators* to conduct complex sequential question answering over a large scale KB. By inserting the previously instantiated variables or KB artifacts (entities, relations, and types in KB) into the operators as arguments, a sequence of operators could form a program like logical forms executable on the KB. At each step the model executes the program and saves the result to memory for use in subsequent steps. For instance, figure 2.2 illustrates how CIPITR maps the query "How many countries have more rivers than China?" into a sequence of actions as designed. Following CIPITR, Stable Sparse Reward based Programmer (SSRP) [14] implements seven *operators* to build programs for answering simple, logical, and quantitative queries. SSRP selects a sequence of operators, which are invoked with intermediate variables as their arguments, to produce the desired programs for question

answering.

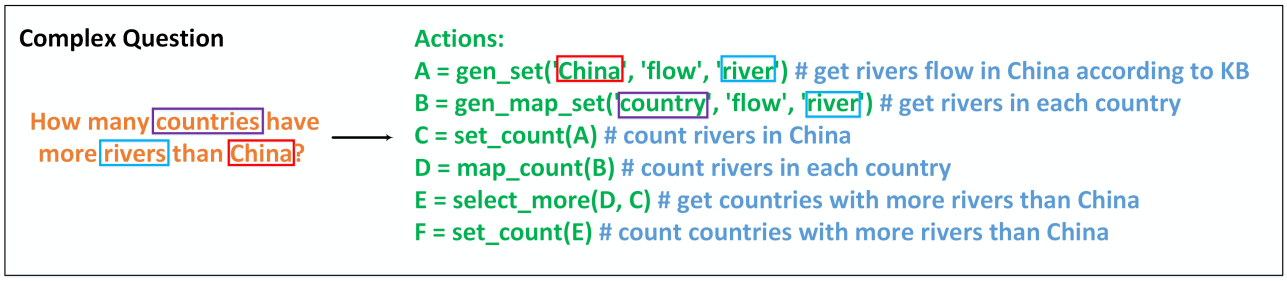

Figure 2.2: An example of a NLQ and the corresponding programs that CIPITR maps into.

To answer sequential questions over a large-scale KB, Dialog-to-Action (D2A) [19] designs a set of *actions* for generating logical forms executable on KB. In their design, the actions are named from A1 to A21. Each action contains three parts: semantic category (set, num, or bool), function symbol (find, count, union, etc.), and a list of arguments. In some cases, the function symbol could be omitted. The argument could be a semantic category, a constant, or an action subsequences. The function of the actions could be categorized into four groups: A1-A3 (to start the logical form), A4-A15 (to operate on sets), A16-A18 (to instantiate KB artifacts or numbers that appear in the query), and A19-A21 (to duplicate a previously generated action subsequence). For example, D2A uses A1 to denote the operation "$start \rightarrow set$", starting the logical form with the semantic category set; expresses action A4 as "$set \rightarrow find(set, r)$", representing the set of entities that connect entity e with relation r; describes the action A15 as "$set \rightarrow \{e\}$" to connect set with entity e. Therefore, D2A maps the question "Who is the president of the United States" to a formal representation "A1, A4, A15, e_{US}, r_{pres}" where e_{US} and r_{pres} represent the entity United States and the relation president_of respectively. Using the similar grammar of the actions defined in D2A, sequence-to-action (S2A) [18], Multi-task Smantic Parsing (MaSP) [69], and a weakly supervised KBQA approach [70] propose several sets of logical forms for complex conversational question answering over KB. Following the action definition in D2A, these methods adapt the actions to the certain complex question answering datasets and solve the question answering tasks.

Other than the KBQA task, there are some lines of works that induce the programs from natural language. Neuro-Symbolic Program Synthesis (NSPS) [106] learns to construct programs with a Domain-Specific Language (DSL). Rothe et al. [107] propose a set of well-formed *programs* based on λ-calculus and LISP [108], and synthesize human questions with the programs. Guu et al. [74] make a formal definition of the *program tokens* used in the SCONE domain[10] and transform natural language

[10]https://nlp.stanford.edu/projects/scone

utterances into executable programs by using the defined tokens. Neural Programmer [20] induces programs with a set of arithmetic and logic *operations* to conduct question answering over tables. A reinforcement learning—based visual question answering method [67], a semantic parser [109] for Cornell Natural Language for Visual Reasoning corpus (CNLVR) [110], and Neural-Symbolic approach for Visual Question Answering (NS-VQA) [111], all develop a set of logic *operations* (which could be extended with set operators and lambda abstraction [109]) for visual question answering task. These three approaches generate a program sequence from the operations to parse the NLQs and execute the programs sequentially.

2.4 KBQA Approaches

The current work synthesizes and extends three mainstream frameworks for KBQA: Information Retrieval-based (IR), template-based, and Neural Semantic Parsing-based (NSP). IR-based approaches represent the questions and candidate answers as low-dimensional vectors. Ranking the candidate answers by matching score or classifying the candidates into positive or negative classes, IR-based methods obtain the answers. Template-based methods resort to predefined templates and artificial rules to conduct semantic parsing and guide the mapping of questions into structured queries. They play a critical role in the KBQA task, simplifying the semantic parsing of questions and decomposing the query utterances. Templates could be devised manually or learned automatically. At the other end of the spectrum, NSP-based research works leverage neural networks for semantic parsing. They use neural models with trainable parameters to construct a semantic parser that fits the given training corpus. A carefully designed neural network architecture and suitable training paradigm are crucial for NSP-based methods to achieve optimal models.

2.4.1 Information Retrieval-based Methods

The IR-based branch first detects the topic entity mentioned in a NLQ and links it to the KB, then extracts the subgraph centered at the topic entity. The extracted subgraph consists of the neighbor nodes within a certain distance (normally a few hops of relations) from the top entity in the KB, where the neighbors in the subgraph are considered the candidate answers. Thus, the IR-based approaches extract features for both the input question and the neighbor nodes in the subgraph, then select the

node among the candidate answers via computing the semantic distance between the question and the candidates. The selected node is treated as the final result of the question. To construct features, several feature engineering approaches and some representation learning methods are introduced.

Yao et al. [5] extract a topic graph in a KB to cover the nodes that are a few hops away from the detected topic entity, viewing the nodes in the topic graph as candidate answers to the input question. They use directed relations and properties as node features for each node in the topic graph. On the other hand, Yao et al. construct a dependency tree for parsing the question, then converting the dependency tree into a question feature graph using the manual-defined rules. They concatenate nodes and edges in the question feature graph to extract question features and further combine them with the node features in the topic graph to produce millions of final features. Also, they run IBM alignment Model 1 [112] to construct relation mappings, i.e., aligning a question with the most likely relation in a KB by mining the corpus ClueWeb[11]. The mappings between the relations in the topic graph and the phrases in the question are also treated as features added to the final features. Finally, a classifier (trained under a machine learning paradigm) uses the final features to determine whether a candidate node in the topic graph is the answer or not.

Methods based on feature engineering are often dependent on pre-defined heuristic rules to extract features, causing limited coverage and may impairing the generalizability. In contrast, representation learning-based methods could project the input questions and candidate answers into the same vector space and convert the KBQA task into a ranking task—the candidate answer that is closest to the question in the vector space is returned as a result. Moreover, to handle complex questions, several methods design frameworks dedicating to multi-hop reasoning.

Bordes et al. [25] first employ representation learning in KBQA task. They first detect the topic entity in a NLQ and then transform the NLQ into a low-dimensional embedding by summing the word vector of each question token. Bordes et al. treat the entities within two hops from the topic entity as candidate answers and thus encode a candidate answer by summing the vectors from three aspects: the candidate answer entity itself, the path between the candidate and the topic entity, and all the entities in KB that connect to the candidate answer entity. They calculate the dot product of a candidate answer's embedding and the question's embedding and thus use the dot products to distinguish the ground-truth answer entities from non-answer entities with the help of a margin-based ranking loss function.

[11]http://lemurproject.org/clueweb09/

Several sophisticated deep neural networks are leveraged to better represent the questions, KB subgraphs, or candidate answers. Dong et al. [113] propose multi-column convolutional neural networks (MCCNNs) to construct representations for questions from three aspects, i.e., answer path, answer context, and answer type. MCCNNs use three convolutional neural networks to map a NLQ into three different question embeddings and then embed a candidate answer by three vectors, considering different aspects, namely answer path (the path between the topic entity and the candidate entity in KB), answer context (the 1-hop entities and relations connecting to the answer path), and answer type (the type embedding matrix derived from Freebase [1]). Dong et al. then compute the dot products (multiply the three question embeddings with candidate answer vectors, respectively) and add the three dot products together to yield a final score. The final score is used to distinguish answer entities from other non-related entities. MCCNNs capture more semantic information in that the networks consider three different aspects to analyze the candidate answer entities, taking full advantage of contextual information in the underlying KB.

The above representation learning-based methods focus on the representation of the candidate answers. In contrast, Hao et al. [114] highlight the representation of the questions. They propose a cross-attention-based neural network to generate question representations with regard to varying candidate answers dynamically. Hao et al. encode candidate answers by four vectors, inspecting the candidates from four aspects: answer entity, answer path, answer type, and answer context. They leverage the attention mechanism to learn the correlation (aka attention weight) between each question token and each answer vector and use the attention weights to generate question representations corresponding to various candidate answer entities dynamically. The similarity scores (i.e., inner products) between the dynamic question representations and the candidate answer representations are used for recognizing the gold answers.

Previous methods don't design a specific method to treat complex questions. Instead, they resort to a unified framework to answer simple and complex questions simultaneously. Modern KBQA systems attempt to design a multi-hop reasoning framework specialized to handle complex questions.

Saxena et al. propose EmbedKGQA [115], a KG embedding method that performs well in answering multi-hop questions over sparse KBs. EmbedKGQA employs a tensor factorization approach, aka ComplEx [116], to project the relations and entities in a KB into a complex space, and utilizes a feed-forward neural network (which is built upon RoBERTa [117]) to embed a NLQ. It then adds the detected topic entity's embedding and the question's embedding together and finds the closet

entity in the complex space as the returned answer. In this process, EmbedKGQA learns the question embedding by optimizing the feed-forward neural network's parameters.

Also, memory network is another important branch of representation learning-based methods. Memory networks are integrated frameworks devoted to storing knowledge with a long-term memory component, making them intrinsically suitable for multi-step reasoning [118]. Memory networks correspond to a sequential reasoning architecture, wherein the writable memory component is used to store and manage long-term facts related to the knowledge in a specific domain (for example, KB triples) [7, 119]. The networks accept a natural language utterance as the input and perform stepwise reasoning by iteratively update the representation of the input by attending to the facts stored in memory. At the last step, the final representation is used to predict the answers to accomplish reasoning.

Miller et al. [120] propose Key Value-Memory networks (KV-MemNN) to bridge the gap between structured knowledge source (for instance, a KB) and unstructured information source (i.e., reading documents) and perform question answering. KV-MemNN designs a key-value structured memory component. The model could use keys to address the facts in memory relevant to the input question and return the corresponding values for stepwise reasoning. Therefore, the knowledge from different sources could be encoded in the same memory. In contrast, the key and value are encoded in different ways, where the key's patterns are designed to facilitate matching the question while the value's features are learned in a way that helps it match the answers. When performing reasoning, KV-MemNN learns to retrieve memories relevant to the input question by key hashing and then computes a normalized relevance weight for each retrieved memory by the inner product between the key and the question representation. Viewing the relevance weights as the weights of the corresponding values, KV-MemNN makes a weighted sum of values, indicating an intermediate reasoning state. The intermediate reasoning state is used to update the question representation. The question representation is repeatedly updated, and the final state is used to predict the answers.

Furthermore, Chen et al. [121] extend the memory network by considering the interaction between the questions and the KB triples. They design a bidirectional attentive memory network (BAM-net) model, using the attention mechanism to measure the correlation (aka attention weights) between a question and relevant KB triples. The attention weights are thus utilized to update the question representations, retrieving the facts more precisely.

Most relevant to our work, Saha et al. [13] propose a CQA model that combines Hierarchical Recurrent Encoder-Decoder (HRED) with a KV-MemNN and predicts the answer by attending to the stored memory. The model first encodes an input sentence with the contextual information into a vector and then utilizes the KV-MemNN to iteratively update the sentence's representation by retrieving and attending the most relevant facts stored in the memory. Then the retrieved memory is decoded to select an answer from all the candidate words as the returned result to the input question.

To sum up, IR-based methods do not rely on hand-crafted rules to analyze the questions and can be trained end-to-end. However, they cannot effectively perform complex question answering and always produce uninterpretable answers—the models could not explain how the answers are generated, making them weak in *interpretability*. Therefore, our research study focuses on designing an interpretable method and can solve complex questions.

2.4.2 Template-based Methods

Template-based methods conduct semantic parsing by mapping the question constituents to corresponding components in query templates. Such methods assume that an input utterance could be paired with a query template to express the query's intention. Usually, the KB artifacts that appear in the question are extracted and inserted into the certain *placeholders* in the template to generate a logical form. By executing the logical form, the answer is derived. Therefore, the semantic parsing task is simplified as mapping the questions into templates and filling the templates with the extracted KB artifacts. A benefit of the template-based methods is that the templates' construction is traceable. The templates could explain how the answers are generated, therefore making the answers *interpretable*.

Berant et al. [4] establish a lexicon to align the natural language phrases with predicates in a KB, then propose a bridging operation to generate additional predicates by synthesizing adjacent predicates. After that, they construct derivations recursively with predicates recognized by the lexicon, the synthesized predicates, and several composition rules. They then train the semantic parser to maximize the probability of generating the correct logical form over all possible derivations. Aqqu [122] proposes three templates to parse the questions. It identifies entities and relations mapped to the input utterance by literal matching, distant-supervised extraction, or supervised recognition. Then it instantiates placeholders of the three templates with the identified KB artifacts to generate matched

query candidates. Based on hand-crafted features, Aqqu ranks the candidates to disambiguate entities and relations jointly. The candidate with the highest-ranking score is finally returned as the structural query that maps the given question. Zhu et al. [84] propose a KBQA system to construct a KB subgraph corresponding to the question. They parse the given questions to construct constituent trees and propose several hand-crafted topological patterns that capture the relationship between the arguments in constituent trees. Then, they detect the entities that a question links to and traverse the paths starting at the detected entities according to the topological patterns. The path with the highest-ranking score is returned as the one leading to the answer node. Hu et al. [123] propose a semantic query graph to match the intention of a question and obtain the answers by mapping the semantic query graph with a suitable subgraph of a KG. To establish the query graph, they propose a relation-first framework to extract the relations that appear in the questions and view the relations as edges in the corresponding semantic query graph. A node-first framework is proposed first to anchor the entities or types to the nodes, then connect them with edges to form the query graph. After building the query graph, they attempt to find a subgraph of the relevant KG that best matches the semantic query graph to resolve the ambiguity of phrase linking (link natural language phrases to entities/relations in KG) and query graph structure jointly.

The KBQA systems that resort to hand-crafted templates could accurately derive the correct answer and explicitly explain generating the answer when the given question conforms to predefined question patterns. However, such systems might yield wrong answers when meeting the questions that are represented in patterns different from the predefined templates. Such a problem makes the systems have limited coverage on questions. Besides, the hand-crafted templates are time-consuming to design, which impair the generalization ability of the KBQA systems. To design a system that can be fit in more different domains, several strategies that could learn templates from corpus automatically or semi-automatically are presented.

QUINT [124] is a KBQA system that could automatically learn the templates from question-answer pairs (QA pair). QUINT detects entities in a question and extracts a query template, i.e., a subgraph from the KG that only contains the detected entities and the relations between them. QUINT also constructs a question template, a dependency parse tree built based on the dependency parse of the question. Therefore, QUINT assumes each query template could be paired with a question template. During inferring, QUINT first builds a parser tree and derives the mapped query template, then instantiates the query template with KB artifacts to generate multiple candidate queries. Ranking

the candidate queries with a random forest classifier, QUINT chooses the highest-ranking query to yield the answer. Other than QUINT, Zheng et al. [125] also propose a method that automatically builds many templates to convey the intention of the input query. Zheng et al. assume each triple in KG could be mapped to multiple natural language patterns in a text corpus and thus build a set of predefined binary templates by the mappings. To represent the meaning of the question, several binary templates are automatically selected from the template set and assembled to generate a complicated template, i.e., a semantic dependency graph. The semantic dependency graph is thus translated into the structured query to obtain the answer. Instead of aligning utterance with the graph structure, Cui et al. [126] automatically learn 27 million templates over 2782 intents from the QA pairs in corpora to solve binary factoid questions (BFQ) and complex questions that are composed of BFQs. In the offline procedure, Cui et al. change the semantic parsing problem into a probability estimation task where an expectation-maximization (EM) algorithm is applied to estimate parameters in a probabilistic model. This statistical model is used to compute the probability distribution of the predicates mapped to a template. In the online procedure, they decompose the complex question into a series of BFQs, derive the templates corresponding to BFQs, use the probabilistic model learned in the offline procedureto infer the relevant predicates, and thus obtain the answers.

Though the methods based on templates possess high interpretability, a practically robust premise for such methods to conduct question answering is sufficient well-defined rules or templates constructed manually or automatically. Such a premise limits the coverage of these methods, which makes them difficult to transfer to different corpora. Therefore, our research study aims to design a method that gets rid of the predefined templates and can be trained end-to-end with full supervision or weak supervision, which applies to more KBQA tasks.

2.4.3 Neural Semantic Parsing-based Methods

Compared with traditional IR-based or template-based methods, instead of using hand-crafted rules or templates, NSP-based methods conduct semantic parsing to translate the NLQs into executable logical forms by harnessing a deep neural network. The generalizability and scalability of the neural networks make NSP-based more effective in performing CQA. These methods usually map the NLQs into abstract meaning representation (e.g., query graphs and trees) and further convert them into structured queries mentioned in 2.3.

The NSP-based methods are trained in an end-to-end manner, where backpropagation learning is employed for calculating the gradient of a loss function with respect to each weight in a neural network model. The gradients are then used by an optimization algorithm (normally stochastic gradient descent (SGD) or its variant optimization algorithm) to update the model's weights for minimizing a target loss function of a predictive model on a given training dataset. The NSP-based methods are task-oriented—designing a specific predictive model for a certain semantic parsing task and choosing a particular loss function accordingly. Also, the given training data leads to two learning paradigms: supervised learning (aka imitation learning, behavior cloning, or teacher forcing)—where a dataset provides pairs of NLQs and annotated logical forms for training, and weak-supervised learning (aka distant-supervised learning)—where pairs of NLQs and corresponding ground-truth answers are supplied for optimizing the neural semantic parsers.

Query Graph. A branch of NSP-based methods resorts to *query graphs* to express a question's intention, while attempting to align the question's semantic relation topological structures with a subgraph in a KB.

Reddy et al. [127] propose GraphParser to perform semantic parsing to query Freebase with natural language utterance, where neither annotations nor denotations are required. GraphParser attempts to represent a question's meaning and depict the question's structure by constructing a semantic graph, which could be mapped to KB, converting the QA task into a graph matching problem. GraphParser employs CCG parser [128] to obtain the semantic parse of a NLQ and transform it into an *ungrounded graph*. The ungrounded graph is mapped to all possible *grounded graphs* in Freebase by replacing the uncertain edges and nodes in the ungrounded graph with the relevant KB artifacts. Each grounded graph corresponds to a unique structured query, which could lead to a predicted answer. The comparison between the predicted answers and the ground-truth answers is treated as a weak supervision signal to train a beam search procedure to recognize the best grounded graph.

Inspired by CCG, Yih et al. [8] propose Staged Query Graph Generation (STAGG) to construct query graphs, which could directly map to structure queries, aka λ-calculus. STAGG leverages a core inferential chain as the core part of a query graph and expands the inferential chain to multiple query graphs by adopting heuristic rules. Viewing the detected topic entity as the root, STAGG explores the paths in KB that link to the topic entity within one hop or two hops (when the middle existential

variable can be grounded to a CVT node[12]) and treat such paths as candidate core inferential chains. STAGG then employs a Siamese neural network [129] (which is constructed on a convolutional neural network (CNN) framework [130, 131]) to identify the core inferential chain among all the candidates by the semantic similarity between a candidate chain and the NLQ. After the core inferential chain is determined, the constraint nodes are attached to the core inferential chain to expand the query graph. At each step, STAGG chooses an action to enlarge the query graph and construct features to rank the graph's state. The high-ranking query graphs are saved for the next step, and other infeasible query graphs are filtered out. At the last step, the final query graph is returned for producing the structured query. STAGG employs a one-layer neural network model built on lambda-rank [132] to construct the ranker for filtering query graphs.

Bao et al. [133] extend the framework of STAGG by covering six more complex constraints, proposing Multiple Constraint Query Graph (MultiCG). The additional complex constraints include multi-entity (more than one entity appear in the question), type constraint (the answer should belong to a type), explicit temporal constraint (the answer should satisfy a temporal requirement), implicit temporal constraint (a question contains implicit temporal expressions), ordinal constraint (the answer should go following a certain ranking), and aggregation constraint (need executing the aggregation operations on a set to obtain the answer). By adding such constraints, MultiCG is more suitable for solving complex questions.

Yu et al. [9] propose a framework dedicated to detecting the relations that appear in a NLQ to enhance the KBQA performance. They design a Hierarchical Residual BiLSTM (HR-BiLSTM) model to detect relations, where a residual learning method is employed to augment the bidirectional hierarchical LSTM network. Yu et al. first detect the topic entity in a NLQ and treat all relation paths related to the topic entity as candidate relation paths. They compare the semantic similarity between the question and all the candidate relation paths from two different granularity, namely relation-level (a phrase of the relation is treated as an input element to the LSTM network) and word-level (every single word of a relation phrase is viewed as an input element). The two different representations are combined to represent the relation paths, while the output of another LSTM network is used to represent questions. The relation path that has the highest cosine similarity with the question representation is returned to obtain answers.

[12]A compound value type (CVT) node is a kind of artificial nodes in Resource Description Framework (RDF) to hold a complex relationship and maintain N-ary facts.

Previous methods concentrate on constructing query graphs by solving entity linking and defining constraint attaching rules, while another line of approaches places more emphasis on designing effective score functions. Luo et al. [134] consider the representation of the questions and query graphs from two aspects: local and global perspectives. They employ a staged generation method to construct candidate query graphs for a NLQ and split the query graphs into several semantic components. They also produce a semantic parse for the question by adopting dependency parsing and extract the dependency path from the wh-word to the question's entity mention in the parse as the local representation of the question. A bidirectional GRU accepts the tokens in the dependency path as the input sequence to generate output vectors. The output vectors are combined by using a max-pooling operation to produce the local representation of a question. Meanwhile, the entire question is fed to another bidirectional GRU to produce global representation. The local and global representations are added together to produce the final representation of the question. On the other hand, the query graph's different semantic components are encoded in the same way as a question's representation. Finally, the cosine similarity between the question's and query graph's final representations is used to rank the candidate query graphs. The graph with the highest ranking score will be used to yield answers.

The method that converts the ungrounded query graphs into grounded query graphs might mislead the semantic parsing. Instead, Maheshwari et al. [135] propose an alternative approach to conduct semantic parsing. They first analyze a NLQ and generate a list of possible query graphs according to the KB's topological structure, then rank the query graphs with regard to the NLQ. They propose a slot matching model to rank the candidate query graphs considering the structure of query graphs. Maheshwari et al. employ an attention mechanism to compute different representations of a NLQ according to different predicates in candidate query graphs. Therefore, the question's representation is used to rank the query graphs.

It can be found that detecting the topic entity in a NLQ plays a critical role in previous query graph-based approaches. These approaches usually do not achieve satisfying performance on the questions with no explicit topic entity included. To solve this problem, Hu et al. [136] propose a State-Transition Framework (STF), combining the aforementioned GraphParser and STAGG. In the beginning, STF detects all entities, relations, and types that appear in a NLQ and initializes a graph. STF then selects one of the predefined atomic operations, namely connect, merge, expand, and fold, to expand the graph by attaching the detected nodes. Also, the edges are mapped to KB relations

by using a CNN-based relation matching model. Every action will update the graph and change its state, and a reward function is used to measure each state. The query graphs with the higher scores will be saved for further updating, while the ones having lower scores will be eliminated. The final high-ranking query graph is used to produce answers.

STAGG is limited to a small range since it only explores the paths within a fixed hops, making it difficult to solve multi-hop questions with more hops. Some researchers propose frameworks that generate query graphs iteratively. Bhutani et al. [137] align several partial queries with a complex query graph by using STAGG, and use a augmented pointer networks [138] to synthesize the partial queries.

Encoder-Decoder Method. Other than query graphs, several methods resort to trees or high-level programming languages to express NLQs. Dong et al. [101] use an attention mechanism to augment the encoder-decoder model to convert the semantic parsing task into a Sequence-to-Sequence (Seq2Seq) problem. Seq2Seq model takes a NLQ as input, and decode the corresponding logical form as output. Moreover, they propose a Seq-to-Tree model, using a hierarchical tree-structured decoder in the framework, to capture the hierarchical structure of logical forms.

Instead of using a conventional sequence encode, Xu et al. [139] propose an encoder-decoder framework considering the syntactic information. They employ a syntactic graph to represent word order, dependency, and constituency of a NLQ and use a graph encoder to encode the syntactic graph's information. They propose a Graph-to-Seq model, where the graph encoder accepts the syntactic graph as input, and the recurrent neural network (which is augmented by an attention mechanism) decodes the corresponding logical forms.

Chen et al. [140] attempt to leverage graph transformer [141] to represent the query graph when solving the multi-hop question answering problem. They pre-define three graph-level operators in their work, including adding a vertex, selecting a vertex, and adding an edge, as the basic operations to construct an abstract query graph (AQG). They design a graph-transformer-based model to predict the graph-level operations at each time step and generate the AQG by sequentially performing the predicted operations. The AQG is thus used as a constraint to generate the query graph that fully conveys the intention of the input question and leads to the answers.

Most relevant to our research study are two state-of-the-art techniques on complex KBQA: D2A [19]

and Multi-task Semantic Parsing (MaSP) [69]. D2A deploys an encoder-decoder neural network model for a sequence to sequence mapping, i.e., reformulating NLQs into logic forms, which are then executed on KBs to compute answers. Furthermore, D2A incorporates dialog memory management in generating logical forms to enable replication of previously generated action subsequence. It labels all training samples with pseudo-gold actions and trains the model by imitation learning. MaSP jointly optimizes two modules to solve the CQA task, i.e., entity linker (the module used to disambiguate the KB artifacts that the natural language phrases in questions could map to) and semantic parser (the module for composing action sequences corresponding to the input question) based on a multi-task learning architecture, relying on annotations to demonstrate the desired behaviors. MaSP constructs an encoder-decoder framework by leveraging Transformer—the network that employs a multi-head attention mechanism with additive positional encoding [142]—except only two stacked layers are used in designing the network. By employing a two-layer Transformer model, MaSP models the dependencies between tokens and represents the tokens with contextual information.

It is worth noting that D2A and MaSP aim to answer context-dependent questions, where each question is part of a multiple-round conversation. Different from them, in this research study, we consider answering the single-turn questions.

It could be found that Transformer-based approaches are verified to be effective in some CQA tasks. It is worth noting that, in the design of our work, the Seq2Seq model is constructed based on the LSTM model. The Transformer, which may have a better effect in some experimental scenarios [142], is not used in our work. In our work, the dimensions of word embedding and LSTM hidden unit are set to 50 and 128, respectively, and the maximum number of parameters of our model is 1.2 million. However, if Transformer is used, even if only its basic structure is used, there are still 12 layers, 12 multi-heads, and 768-dimensional hidden units in its neural network.The number of parameters in Transformer is about 110 million [143, 144], which is nearly 100 times the parameters of the model designed in our work. Since only a small number of samples are used in our work, and a relatively weaker distant supervision signal (reward signal) is adopted, the model's parameters are harder to optimize. Under such circumstances, the model per se is not easy to converge. Once there are more parameters to train, the model is more difficult to optimize. Therefore, following Occam's razor [145], we choose the LSTM model to build a programmer instead of using Transformer.

The methods based on *Encoder-Decoder framework* have shown great success in several KBQA tasks without relying on hand-crafted templates or rules. However, they achieve state-of-the-art

performance when a large number of *annotations* are available for training. The availability of such large training data has generally been a prerequisite in these methods, presenting a challenge in the KBQA scenarios where annotated labels are expensive to collect.

Reinforcement Learning Method. To alleviate the dependence on annotated labels, Lan et al. [146] propose an RL-based framework to answer complex questions with constraints as well as multiple hops of relations. Lan et al. use a staged query graph generation method to construct a query graph to express questions' intention and find the answers in a KB. They define three types of actions to construct the query graph, i.e., extend, connect, and aggregate actions. They train a policy network under the RL paradigm to decide a stepwise action at each step—to execute the extend, connect or aggregate action to grow the query graph with one edge or one node. At each step, they rank all candidate query graphs by using a 7-dimensional feature vector and keep the most feasible ones for predicting answers. The F1 score of the predicted answers concerning the ground truth answers is treated as the reward signal to train the policy network.

Also, several modern approaches attempt to train the model with *denotations*, i.e., the ground-truth answers, as weak supervision.

When answering multi-hop questions over KG, it is non-trivial for the QA systems to link the topic entity in a question with the KG and further find the correct path which starts with the topic entity and leads to the answer entity in KG. In certain scenarios, since only question-answer pairs are available, the above problems could be even harder while the topic entities and paths are not annotated. To overcome such challenges, Zhang et al. [147] propose a variational reasoning network (VRN), an end-to-end architecture that handles topic entity recognition and multi-hop path reasoning simultaneously. Treating the topic entity as a latent variable, VRN constructs two probabilistic components to model the process of topic entity recognition and path reasoning respectively. Since the values of the latent variables are discrete, Zhang et al. employ the REINFORCE algorithm to train the VRN end-to-end by treating the probabilities derived from the two probabilistic components as reward signals.

To handle multi-hop questions, Das et al. [148] reduce the question answering process to a finite horizon sequential decision making problem and employ a neural reinforcement learning approach to learn to navigate the KG to find the desired paths that lead to the answer to a question. The approach treats the chosen relation in KG as the action and the entity links to the chosen relation as the observation and deploys a LSTM network to encode the action and observation to construct the

history embedding. The history representation is thus used to compute a probability distribution over all the possible relations to determine the subsequent action. Therefore, Das et al. treat the LSTM network as the policy network and employ the REINFORCE algorithm to optimize the policy network by assigning the policy with a positive terminal reward when the final location is the correct answer.

Similarly, Lin et al. [149] also formulate the multi-hop question answering problem as a 'walk-based query answering', namely learning to walk towards the correct answer following the appropriate path in the KG. An on-policy RL algorithm, i.e., REINFORCE is often adopted in the walk-based QA framework to encourage the policy network to walk through the paths that could lead to correct answers. However, Lin et al. find that the spurious paths—the incorrect paths that accidentally reached the correct answers—would mislead the policy network and further impair the policy's performance. To alleviate the spurious problem, they first adopt the KG embedding models to compute a soft reward instead of a binary reward to measure the feasibility of the entity visited in the current time step. Also, Lin et al. propose an action dropout method to randomly mask certain outgoing edges of the currently visited entity to decrease the possibility of sampling a spurious path. By combining the soft reward and action dropout mechanisms, the walk-based QA system could alleviate the spurious problem and outperform existing path-based KGQA models.

Instead of attempting to find the correct paths to the answers, REINFORCE algorithm could also be used in the NPI approaches, where two powerful ideas are married: deep learning for representing the natural language utterances and symbolic program execution for logic reasoning. Liang et al. [11] propose NSM, a Seq2Seq model learned with RL. NSM deals with multi-hop questions with two components: the programmer and the computer. The programmer devotes itself to generating logical forms for each question while the computer executes the logical forms to obtain answers. By employing an EM-like mechanism, NSM iteratively finds the pseudo-gold trials for the training questions. In the RL training process, NSM assigns the pseudo-gold trials with a deterministic probability to anchor the model to the high-reward trials. Then NSM uses a comparison of the computed answers and the ground-truth answers, i.e., the F1 score of the answers, as weak supervision to train the programmer.

In the CQA problem setup, CIPITR [58] translates a natural-language complex question into a multi-step executable program using the NPI technique. CIPITR is the state-of-the-art method for complex question answering tasks. CIPITR does not require gold annotations and can learn from auxiliary rewards, KB schema, and inferred answer types. It employs high-level constraints and additional auxiliary rewards to alleviate the sparse reward problem. By using pre-defined high-level

constraints, CIPITR restricts the model to search for possible actions that are semantically correct. Besides, it rewards the model with additional feedback when the generated answers have the same type of ground-truth answers.

Another NPI-based model, SSRP [14] is proposed to solve complex question answering task. SSRP utilizes an unsupervised joint Entity, Relation, Type (ERT) linking method to identify all the candidate KB artifacts in the queries, then feeds the candidate KB artifacts into an NPI model along with the gold answers. To alleviate the sparse reward problem caused by the noisy ERT linking results, SSRP designs a noise-resiliency wrapper over an NPI model by introducing the concept of a reference programmer. The reference programmer shares the same architecture with the current programmer but uses older version parameters. By comparing the programs and the corresponding rewards obtained by the reference programmer with the current programmer, SSRP controls the extent of backpropagation in training the programmer. Therefore, the programmer could be updated in a more stable fashion, crippling noise and mitigating the sparse reward problem.

Other than REINFORCE algorithm, other lines of RL algorithms could also be applied in the CQA tasks. Stepwise Reasoning Network (SRN) [150] is a neural model based on RL proposed to solve the multi-relation questions over a knowledge graph. SRN formulates the multi-relation question answering problem as a sequential decision problem. The model selects the next action among the candidate actions (consisting of the relations and entities linked to the currently visited entity) at each step. Since the delayed and sparse reward problems are shown to arise from previous RL-based models using non-potential-based rewards, in addition to the final answers, SRN thus employs a potential-based reward shaping strategy to produce additional training rewards to guide the policy when making decisions. In SRN, the potential-based rewards correspond to distance-based and subgoal-based heuristics.

In a similar vein, Qiu et al. [151] extend the traditional Markov Decision Process (MDP) [152] framework to solve the multi-hop question answering over knowledge bases. Qiu et al. convert the QA process into a stepwise decision procedure by designing a Director-Actor-Critic framework. In this framework, the Director selects a high-level strategy (decide the type of triple to be formed), the Actor sequentially performs low-level actions (i.e., triple generation), and the Critic computes the semantic similarity between the newly generated triple and the input question. The hierarchical RL algorithm is employed in the Director-Actor-Critic framework. The sequentially generated triples are used to construct a query graph to translate to executable logical forms directly.

Wang et al. [153] also employ deep reinforcement learning (DRL) based algorithm to tackle multi-step question answering problems. They propose a multi-step coarse to fine question answering (MSCQA) system to handle documents of various lengths. MSCQA consists of three modules, namely sentence selection module, which is used to select sentences from the relevant documents, sub-context generation module, used to remove the spurious predicted answers from the document context, and answer generation module—used to generate answers from the selected sentences. Furthermore, MSCQA defines three actions, each corresponds to one of the three modules respectively, in the DRL framework. MSCQA, therefore, employs the actor-critic-based algorithm to construct the DRL framework, where two neural networks are built to model the actor and the critic. In the DRL framework, the actor network is trained to predict the next step's action by taking the current step's state as the input, while the critic network learns to evaluate the gained value by performing such action.

It is worth noting that compared with other variants of the policy gradient method, existing DRL-based CQA approaches tend to utilize the REINFORCE algorithm when constructing the DRL framework. This inclination is because most QA systems reduce the CQA task to a sequence generation problem where the input is the question, and the output is the corresponding logic forms, meaning representations, or structural queries. Under such circumstances, REINFORCE algorithm, an important policy-gradient reinforcement learning algorithm, is suitable for training the sequence generation model since the algorithm has shown the effectiveness for a variety of sequence generation tasks [154, 155, 156]. Following the previous works, we also introduce the REINFORCE algorithm into our work.

Meta Learning Method. The conventional approach to KBQA is to train one model to fit the entire training set and then use it for answering all complex questions at the test time. However, such a one-size-fits-all strategy is sub-optimal as the test questions may have diversity due to their inherently different characteristics [17]. In some scenarios, the structure, length, difficulty, data distribution, and other characteristics of the questions are quite different. It is often difficult to find a set of the global optimal parameters for the neural network, and it is hard to fit them into the training samples. The QA models might have inconsistent performance if they use a set of parameters to answer different types of questions. The QA models will always have better performance on certain types of questions while performing poorly on other types of questions. Therefore, the researchers introduced the meta-learning method, which changed the optimization goal of the neural network from finding a set of global optimal parameters to finding a set of optimal initial parameters. When

answering new questions, specific parameters suitable for current queries were dynamically generated on the fly with a small number of samples. Multiple sets of parameters could be adaptively generated for multiple types of questions to answer questions accurately. Therefore, meta-learning is introduced to solve the one-size-fits-all problem.

Meta-learning, aka learning-to-learn, aims to make learning a new task more effective based on the inductive biases that are meta-learned in learning similar tasks in the past [157, 158]. Meta-learning is a popular framework for learning an algorithm over a distribution of tasks. The algorithm could produce an adaptive model that performs well on a previously unseen task by learning with limited data sampled from this task [159]. Conventional deep neural networks have shown great success in many applications by being trained with large amounts of labeled data. Still, they tend to struggle when required to adapt quickly to an unseen task with a small amount of training data [160]. Moreover, another challenge of standard deep neural networks resides in learning new tasks incrementally on the fly without forgetting or distorting the previously learned knowledge [161]. Therefore, the monolithic, one-size-fits-all neural networks trained under supervised-learning (i.e., imitation Learning) or weak-supervised learning paradigms (aka RL) always exhibit low performance when training under a limited amount of samples or adapting to varying tasks [160]. In contrast, humans possess the cognitive ability of continuous learning, i.e., rapidly learning new concepts from a few samples by exploiting the prior knowledge and accumulated experience (aka the inductive bias [162]) of previously learned concepts [163], which is the ability that distinguishes human cognition from machine intelligence. Humans could extract the inductive bias from the seen tasks and thus apply it to unseen tasks [164]. For example, people can acquaint themselves with a novel object by observing a few images of that object [165, 166], or learn how to play the Atari game of Frostbite much more quickly [163] than double-dueling-DQN model [167]. In response, meta-learning is proposed to mimic the capacities that distinguish human intelligence from the monolithic neural networks [168], which could extract generic knowledge over the seen tasks and apply it to solve unseen tasks [169, 170].

Meta-learning seeks to search for a high-level strategy that generalizes the previously acquired knowledge to the novel but similar tasks while capturing the specific essence of the novel task [171]. To accomplish this, recent meta-learning approaches design a meta-learner to find the high-level strategy [160]. Instead of being trained to fit a single task—generalizing to unseen data points sampled from this task's data distribution, the meta-learner is trained to discover a rule that generalizes to a broader scope—a distribution of related tasks. Generally, meta-learning can be viewed as constructing

a high-level strategy h by learning from a distribution of tasks D_{tr}. For instance, in a one-shot learning setting, D_{tr} denotes a set of labeled samples categorized into C groups, and h corresponds to a C-way classifier.

A variety of approaches have been proposed for meta-learning. A branch of these methods formulates the meta-learning problem as a two-level framework: a base-level model performing learning the task-specific patterns and a meta-level model acquiring the generic knowledge across tasks [172, 173]. When facing a novel task, the base-level model utilizes the generic knowledge to generalize to the novel task. This line of meta-learning algorithms either integrates the base and meta-level models in a single learner [174], or encodes the high-level strategy in the weights of an additional neural network (for example a LSTM network [175, 176], an RNN network [158], or RL-based policy [177]).

Another branch of the approaches is to learn the optimal initialization of a network's parameters, and thus finetune the parameters when testing a new task [178]. Model-Agnostic Meta-Learning (MAML) [179] is an important thread of this branch of approaches. Observing the initial weights have a substantial effect on models' performance, MAML converts the meta-learning problem into an initial parameter optimization problem—MAML learns a task-agnostic model initialization representing the common patterns across a distribution of tasks, and finetunes the model to adapt to a novel task by a fixed number of gradient descent steps from this initialization. MAML formulates the meta-learning in an episodic manner, wherein an algorithm is trained to minimize the loss of a query sample q by adapting to the support set s^q, which consists of samples extracted from the data distribution of the task that q belongs to. Suppose the initial parameters of the model to be learned is θ, MAML first evaluates the loss $\mathcal{L}_{s^q}$ on the support set s^q, and then updates θ using gradient descent generated by $\mathcal{L}_{s^q}$,

$$\theta' = \theta - \alpha \frac{\partial \mathcal{L}_{s^q}}{\partial \theta} \tag{2.1}$$

where α is step size that could be defined as a hyperparameter or meta-learned.

MAML then updates the initial parameters θ to minimize the loss $\mathcal{L}_q$ incurred on the query sample q by using the adaptive parameters θ',

$$\theta = \theta - \beta \frac{\partial \mathcal{L}_q}{\partial \theta} \tag{2.2}$$

where β is the step size for updating θ.

In 2.2, the derivative could be expanded using the chain rule:

$$\frac{\partial \mathcal{L}_q}{\partial \theta} = \frac{\partial \mathcal{L}_q}{\partial \theta'} \frac{\partial \theta'}{\partial \theta} = \frac{\partial \mathcal{L}_q}{\partial \theta'} \frac{\partial (\theta - \alpha \frac{\partial \mathcal{L}_{s^q}}{\partial \theta})}{\partial \theta} = \frac{\partial \mathcal{L}_q}{\partial \theta'} (\mathcal{I} - \alpha \frac{\partial^2 \mathcal{L}_{s^q}}{\partial \theta^2}) \tag{2.3}$$

In 2.3, it can be found that the MAML meta-gradient update involves computing a gradient through a gradient, i.e., Hessian-vector products, to update the initial parameters, which plays a critical role in sensible gradient-based learning. It is worth noting that empirical studies in [159] indicate that using an approximation to first-order derivatives for meta-gradient update (ignoring second-order derivatives and only considering first-order derivatives) could achieve a competitive performance on several benchmarks and is more convenient for implementation.

Therefore, Reptile [180], as a first-order meta-learning algorithm, is adopted in several meta-learning settings to replace MAML. MAML has a simplified architecture since it does not require an additional network or framework for meta-learning, and thus is the focus of this research study. In Reptile, suppose the initial parameters is θ, the task used in meta learning is τ, and the loss on the task τ is L_τ. The function $SGD(\cdot)$ is used to compute the gradient on loss L_τ and update the parameters starting from θ,

$$W_\tau = SGD(L_\tau, \theta, k) \tag{2.4}$$

where k denotes the function has performed k gradient steps.

Reptile then update the initial parameters θ as:

$$\theta \leftarrow \theta + \epsilon(W_\tau - \theta) \tag{2.5}$$

where ϵ denotes the step size for updating θ.

The equation 2.5 is inspired by the following heuristic idea: suppose $\mathcal{W}_\tau^*$ is a set of theoretical optimal parameters to solve the task τ, then the objective of the algorithm Reptile is to find a parameter θ in the parameter space that is the closest to one certain optimal parameter in the set $\mathcal{W}_\tau^*$. Therefore, the objective function is designed as:

$$\min_\theta \frac{1}{2} D(\theta, \mathcal{W}_\tau^*)^2 \tag{2.6}$$

In the equation 2.6, $D(\theta, \mathcal{W}_\tau^*)^2$ denotes the squared Euclidean distance between a single point

θ and a set of points $\mathcal{W}_\tau^*$. The gradient of the squared Euclidean distance $D(\theta, \mathcal{W}_\tau^*)^2$ is the vector $2(\theta - p)$, where p is the closest point in $\mathcal{W}_\tau^*$ to θ. To minimize equation 2.6, the gradient could be computed as:

$$\nabla_\theta[\frac{1}{2}D(\theta, \mathcal{W}_\tau^*)^2] = \theta - p \tag{2.7}$$

The parameter θ could be updated after performing a stochastic gradient update on a task τ as:

$$\theta \leftarrow \theta - \epsilon\nabla_\theta[\frac{1}{2}D(\theta, \mathcal{W}_\tau^*)^2] = \theta - \epsilon(\theta - p) = \theta + \epsilon(p - \theta) \tag{2.8}$$

In the equation 2.8, we use W_τ to approximate p, i.e., the point in $\mathcal{W}_\tau^*$ that is the closet to θ. Therefore we could derive the equation 2.5 by replacing p in the equation 2.8 with W_τ. Reptile reduces the computational cost, and it is verified to achieve the same effect as MAML in many practical tasks. Therefore it can replace MAML in many scenarios.

In the NLP spectrum, MAML has been employed in many different works, including training neural machine translation (NMT) systems for low-resource languages [181], learning personalized chatbots [182], and learning a neural semantic parser with limited rules while neither annotations nor denotations are used [183].

More relevant to this research study, in question answering settings, Huang et al. [17] propose a relevance function to find similar samples to form a pseudo-task for each WikiSQL [184] question. Subsequently, they reduce a supervised learning problem into a meta-learning problem and employ MAML to adapt the programmer to each pseudo-task. More closely related to our work, Guo et al. [18] propose Sequence-to-Action (S2A) by using MAML to solve CQA problems. They label all the examples in the training set with pseudo-gold annotations, then train an encoder-decoder model to retrieve relevant samples and a Seq2Seq based semantic parser to generate actions based on the annotations. Based on the retriever's similar samples, S2A establishes a meta-learning task for each question and thus employs MAML to finetune the programmer. It is worth noting that S2A aims to answer conversational questions, while in this study, we consider answering single-turn questions. Also, unlike S2A, we introduce a Meta-RL approach in this research study, which uses RL to train an NPI model without annotating questions in advance.

Although the above NSP-based methods can enlarge the coverage of questions, the availability of sufficient gold logical forms has generally been a prerequisite in training a neural semantic parser. Also,

the one-size-fits-all problem impairs the existing methods' performance when faced with the KBQA questions that vary vastly. Therefore, in this research study, we aim to design a model that could be trained without the restriction of the gold annotations, alleviate the sparse reward and data-inefficiency problems that inherently exist in the RL-based methods, and could solve the one-size-fits-all problem.

2.5 Summary

To sum up, in this research study, we aim to solve the CQA task that automatically finds precisely the piece of information in a KB to answer complex questions raised by users. To accomplish this task, we need to (i) understand the NLQs raised by users, (ii) retrieve relevant facts in KB, and (iii) conduct logical or arithmetic reasoning over the retrieved facts to infer the answers. It is worth noting that the KBQA subtasks, i.e., *Entity Linking* and *Predicate recognition* are necessary for understanding the questions' intention, where the accuracy of these tasks have a significant influence on the performance of the entire CQA system. Modern CQA systems either incorporate independent components specifically designed for KBQA subtasks into their pipeline framework or combine the subtasks and the answer-inference process into a unifying deep learning framework. Moreover, there are several branches in question answering, including IR-based, template-based, and NSP-based approaches. IR-based methods project the questions and candidate answers into the same vector space and treat the answer inference process as a classification or ranking task. However, there exists a gap between the natural language utterances and the returned answers in IR-based methods in that it is hard to explain why users receive such answers. In contrast, template-based methods bridge this gap by introducing templates, a custom-defined pattern representing a query's structure and intention, simplifying the semantic parsing of input utterances. Template-based methods map a question into templates and then translate the templates into structured queries to obtain answers. Though template-based methods can generate interpretable answers, such methods are highly dependent on pre-defined templates, resulting in limited coverage. To improve CQA models' generalizability, as a critical branch of CQA, NSP-based approaches design a neural network to conduct semantic parsing for answering a question. This line of approaches generally achieve better performance than other branches; however, they still face a number of challenges, including the limitation of manual annotation and the one-size-fits-all problem.

It can be found that though existing work could achieve competitive results in answering complex

questions, there remain several major challenges that need to be handled. Therefore, we propose our approach that develops a neural-symbolic framework to solve the CQA task while addressing these challenges. To handle complex questions and improve interpretability, we leverage the NPI method. To design a CQA framework by only using denotations while alleviating the sparse reward and data inefficiency problems that inherently lie in the conventional NPI models, our research study presents an RL architecture that incorporates curriculum learning and memory-buffer. Furthermore, to learn an adaptive model to precisely answer each novel question instead of using a one-size-fits-all model, we upgrade the RL model into a meta reinforcement learning (meta-RL) framework, where the model is trained to adapt to the target question quickly. To improve the meta-RL model's performance, we design an optimal retriever to find the most appropriate instances for the target question. To unify the retriever and the meta learner's learning process, we propose a new learning algorithm that jointly optimizes both models. The specific details of our approach will be described in the following chapters.

Chapter 3

Neural-Symbolic Complex Question Answering Over Knowledge Bases

KBQA automatically answers users' NLQs by understanding the queries, exploring KB facts, and returning the relevant facts as answers. The simple question is a type of questions that refer to a single fact, for instance, a triple (China, capital, Beijing) could be used for answering the query "What is the Capital of China?". However, in practice, people tend to formulate questions with more complexity—the complex questions that are not limited to a single triple—to meet their ever-growing information needs. Therefore, the task of retrieving answers based on more than one triple in a KB, aka CQA, is an active research area and attracts significant attention recently.

In comparison with simple questions, complex questions typically involve aggregating information from multiple KB facts, conducting logical reasoning or arithmetic operations in order to infer the answers. Specifically, this kind of questions includes the following types: (i) questions need sequential (multi-hop) reasoning over a chain of triples, e.g., "What university did president of the United States graduate from?", (ii) questions with certain constraints, e.g., "Who is the president of the European Union 2012?", and (iii) questions require arithmetic processing, e.g., "How many countries have more rivers than China?". To cope with these complex questions, a structured query that correctly assembles multiple functions is required and is the main subject of this research study.

To circumvent the CQA problem, several IR-based KBQA systems are proposed. Such retrieval models are either dependent on traditional lexical term-based search—which performs phrasal match-

ing instead of understanding the meaning of user queries [5], or dense representation method—which projects queries and KB nodes into a vector space and finds the most aligned vector with a given question vector to complete the answer retrieval [25, 81, 113]. However, the IR-based methods lack interpretability because they cannot explain the answer inference process; therefore, they cannot be applied in specific settings where explanations are required.

In contrast, numerous template-based works [124, 126] rely on parsing a question to construct templates that reflect the syntactic structure in the utterance and represent the question intention. These methods decompose a complex question into multiple simple subquestions, map the subquestions into pre-defined templates according to artificial rules, and instantiate the templates with KB artifacts identified in the question [125]. However, such template-driven approaches have strict limitations in coverage because they cannot handle the questions whose structure and format differ from the training samples in a given dataset.

To improve generalizability, the end-to-end models, including feed-forward [185] and encoder-decoder networks [101, 139] are introduced to produce logical forms for each input question. Being built upon the imitation learning paradigm, these methods either depend on given semantic parse labels [10] or pseudo-gold annotations that collected by a brute-force search method [19, 69] to train the networks. However, it is widely believed that collecting such task-specific annotations can be a time-consuming task [186].

Instead of using the behavior cloning method, several state-of-the-art NPI-based KBQA systems [14, 58] use final answers as weak supervision to train the end-to-end models in the cases that only NLQs and the corresponding answers are available. These end-to-end models learn to compose a sequence of tokens that forms the logical form by generating one token at a time, conditioning on previously generated tokens or intermediate execution results [11]. Such methods divide the generation process into multiple steps and use the information already at hand to choose a valid subsequent token iteratively. Therefore, the search space of the logical forms would grow exponentially with each step and blow up to a very large size eventually [14, 187]. Such a huge search space makes NPI models suffer from sparse reward problem, where out of all possible programs induced by a model, only a tiny set of them could yield non-zero rewards for training. Also, a data-inefficiency problem—that excessive trials are needed for training the model to fit a particular question—arises from the search space's exponential size.

To solve the above challenges in the CQA task, our proposed CQA system should:

1. use custom-defined actions to generate logical forms for answering a complex question,
2. explain the process of answer inference,
3. solve the questions independent of pre-defined templates or rules,
4. be trained end-to-end without using annotations,
5. augment the sparse extrinsic reward, and
6. improve the sample efficiency.

The relevant CQA methods have been deeply analyzed in the attached research paper's literature study. Since existing CQA systems that satisfy all the aforementioned requirements are not prevalent, a novel NPI-based CQA method is proposed. Conducting experiments on several CQA datasets, our method is proved to outperform the existing state-of-the-art CQA approaches. We name the model as **N**eural-**S**ymbolic **C**omplex **Q**uestion **A**nswering (**NS-CQA**). This research work has been published in the *Journal of Web Semantics 2020*. The complete version of the manuscript is attached in the subsequent pages.

Hua, Y., Li, Y., Qi, G., Wu, W., Zhang, J. and Qi, D. Less is more: Data-efficient complex question answering over knowledge bases. In *Journal of Web Semantics, 2020*, 65, p.100612.

Web Semantics: Science, Services and Agents on the World Wide Web 65 (2020) 100612

Contents lists available at ScienceDirect

Web Semantics: Science, Services and Agents on the World Wide Web

journal homepage: www.elsevier.com/locate/websem

Less is more: Data-efficient complex question answering over knowledge bases

Yuncheng Hua [a,d], Yuan-Fang Li [b], Guilin Qi [a,c,*], Wei Wu [a], Jingyao Zhang [a], Daiqing Qi [a]

[a] School of Computer Science and Engineering, Southeast University, Nanjing, China
[b] Faculty of Information Technology, Monash University, Melbourne, Australia
[c] Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education, Nanjing, China
[d] Southeast University-Monash University Joint Research Institute, Suzhou, China

ARTICLE INFO

Article history:
Received 3 October 2019
Received in revised form 4 May 2020
Accepted 2 September 2020
Available online 16 October 2020

Keywords:
Knowledge base
Complex question answering
Data-efficient
Neural-symbolic model
Reinforcement learning

ABSTRACT

Question answering is an effective method for obtaining information from knowledge bases (KB). In this paper, we propose the Neural-Symbolic Complex Question Answering (NS-CQA) model, a data-efficient reinforcement learning framework for complex question answering by using only a modest number of training samples. Our framework consists of a neural *generator* and a symbolic *executor* that, respectively, transforms a natural-language question into a sequence of primitive actions, and executes them over the knowledge base to compute the answer. We carefully formulate a set of primitive symbolic actions that allows us to not only simplify our neural network design but also accelerate model convergence. To reduce search space, we employ the copy and masking mechanisms in our encoder–decoder architecture to drastically reduce the decoder output vocabulary and improve model generalizability. We equip our model with a memory buffer that stores high-reward promising programs. Besides, we propose an adaptive reward function. By comparing the generated trial with the trials stored in the memory buffer, we derive the curriculum-guided reward bonus, i.e., the proximity and the novelty. To mitigate the sparse reward problem, we combine the adaptive reward and the reward bonus, reshaping the sparse reward into dense feedback. Also, we encourage the model to generate new trials to avoid imitating the spurious trials while making the model remember the past high-reward trials to improve data efficiency. Our NS-CQA model is evaluated on two datasets: CQA, a recent large-scale complex question answering dataset, and WebQuestionsSP, a multi-hop question answering dataset. On both datasets, our model outperforms the state-of-the-art models. Notably, on CQA, NS-CQA performs well on questions with higher complexity, while only using approximately 1% of the total training samples.

1. Introduction

Knowledge base question answering (KBQA) [1–5] is an active research area that has attracted significant attention. KBQA aims at interpreting natural-language questions as logical forms, action sequences, or programs, which could be directly executed on a knowledge base (KB) to yield the answers.

Many techniques have been proposed for answering single-hop or multi-hop questions over a knowledge base. Neural network-based methods [1–5] represent the state-of-the-art in KBQA. More recently, complex knowledge base question answering (CQA) [6] has been proposed as a more challenging task. Complex question answering, the subject of this paper, focuses on aggregation and multi-hop questions, in which a sequence of discrete operations – e.g., set conjunction, counting, comparison, intersection, and union – needs to be executed to derive the answer.

Complex question answering is typically cast as a *semantic parsing* problem, whereby natural-language questions are transformed into appropriate structural queries (sequences of discrete actions). Such queries are then executed on the knowledge base to compute the answer. Consider the complex question "How many rivers flow through India **and** China?" as a motivating example. Fig. 1 shows an incomplete sub-graph relevant to this question. To answer this question, all entities whose type is "river" and link to the entity "China" with edge "flow" will first need to be retrieved from the KB to form the candidate set S_A. Meanwhile, the candidate set S_B will also be formed to represent those rivers that flow through India. After obtaining the intersection of S_A and S_B, the number of elements in the intersection can finally be identified as the correct answer to the question. It

* Corresponding author.
E-mail addresses: devinhua@seu.edu.cn (Y. Hua), yuanfang.li@monash.edu (Y.-F. Li), gqi@seu.edu.cn (G. Qi), wuwei@seu.edu.cn (W. Wu), zjyao@seu.edu.cn (J. Zhang), daiqing_qi@seu.edu.cn (D. Qi).

https://doi.org/10.1016/j.websem.2020.100612
1570-8268/

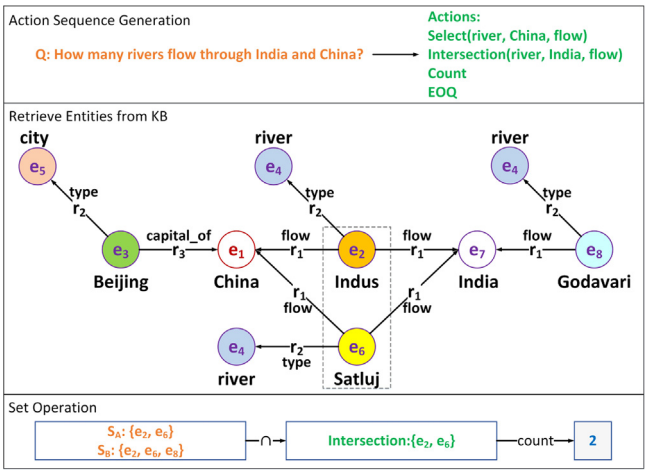

Fig. 1. An example illustrating the task of complex question answering.

can be seen that a diverse set of operations, including selection, intersection, and counting operations need to be sequentially predicted and executed on the KB.

Sequence-to-sequence (seq2seq) models learn to map natural language utterances to executable programs, and are thus good model choices for the complex question answering task. However, under the supervised training setup, such models require substantial amounts of annotations, i.e., manually annotated programs, to effectively train. For practical KBQA applications, gold annotated programs are expensive to obtain, thus most of the complex questions are not paired with the annotations [7]. Reinforcement learning (RL) is an effective method for training KBQA models [8,9] as it does not require annotations, but only denotations (i.e. answers) as weak supervision signals. However, RL-based KBQA methods face a number of significant challenges.

Sparse reward. Neural-symbolic models that are optimized through RL have been proposed for the complex question answering task [7–9]. In the RL context, questions of the same pattern could be regarded as one single *task*, while programs trying to solve these similar questions are considered *trials*. Instead of using the gold annotations, neural-symbolic models employ *rewards*, i.e., comparisons between the predicted answer and the ground-truth answer as the distant supervision signal to train the policy [8,9]. Usually, a positive reward could only be given at the end of a long sequence of correct actions. However, in the initial stage of model training, most of the trials sampled from the sub-optimal policy attain small or zero rewards [10]. We randomly selected 100 questions from a complex question answering dataset, and manually inspect the generated trials. We found that more than 96.5% of the generated trials led to wrong answers and got zero reward. Thus, this **sparse reward** problem in the complex question answering task is a major challenge that current neural-symbolic models face.

Complex Imperative Program Induction from Terminal Rewards (CIPITR) [7] is the state-of-the-art method for the complex question answering task. It employs high-level constraints and additional auxiliary rewards to alleviate the sparse reward problem. By using pre-defined high-level constraints, CIPITR restricts the model to search for possible actions that are semantically correct. Besides, it rewards the model by the additional feedback when the generated answers have the same type of ground-truth answers. However, on the one hand, defining the high-level constraints comes at the cost of manual analysis to guide the model's decoding process, which is tedious and expensive. On the other hand, CIPITR only harnesses the type of predicted answers as the auxiliary reward, which makes many failed experience still only have a zero reward. At the initial stage of training, most of the programs generated by CIPITR fail to yield expected answers and gain zero rewards, thus most of the programs do not contribute to model optimization. Thus, the sparse reward problem remains a challenge for CIPITR.

Data inefficiency. Furthermore, due to the sparse supervision signals, such models are often **data inefficient**, which means many trials are required to solve a particular task [11]. Being trained from scratch, RL models often need thousands of trials to learn a simple task, no matter what policy is employed to search programs. When faced with a large number of questions, training such models would consume an enormous amount of time. One way to increase data efficiency is to acquire task-related prior knowledge to constrain the search space. However, in most cases, such prior knowledge is unavailable unless manual labeling is employed. Hence, the data inefficiency problem often makes models expensive to train and thus infeasible/impractical. Liang et al. [8] proposed the Neural Symbolic Machine (NSM) that maintains and replays *one* pseudo-gold trial that yields the highest reward for each training sample. When using RL to optimize the policy, NSM assigns a deterministic probability to the best trial found so far to improve the training data efficiency. However, in NSM, the best trial to be replayed might be a spurious program, i.e., an incorrect program that happens to output the correct answer. Under such circumstances, NSM would be misguided by the spurious programs since such programs could not be generalized to other questions of the same pattern. Besides, NSM only harnesses the accuracy of the predicted answers to measure the reward, hence also suffers from the sparse reward problem.

Large search space. Large KBs, like Wikidata [12] or Freebase [13], contain comprehensive knowledge and are suitable to be sources for complex question answering. The KBs with smaller size usually embrace a limited number of facts, thus are insufficient to yield the required answers. To answer the questions, searching for KB artifacts (entities, classes, and predicates in KB) that are related to the questions is a crucial step. However, large KBs often lead to vast search space. Given a sequence of actions, at each step of its execution, both the operator and the parameters, i.e., KB artifact, used in action need to be correct. To produce the correct result, the actions in the sequence also need to follow a particular order. With the above factors combined, searching desired action sequences would consume a considerable amount of time and memory, which makes the training process slow and expensive. How to design a set of simple yet effective actions that could reduce the search space remains a challenge.

In this paper, we propose a Neural-Symbolic Complex Question Answering framework, NS-CQA. It trains a policy to generate the desired programs from the comparison between the generated answer and the ground-truth result. We incorporate a memory buffer, design an adaptive reward function, and propose a curriculum-guided reward bonus to improve the model.

To **reduce search space**, we use a combination of simple yet effective techniques to reduce model complexity, increase generalizability, and expedite training convergence. We employ the widely-used masking technique to allow the model to handle unseen KB artifacts effectively. In conjunction, the copy mechanism [14] is also employed to reduce the size of the decoder output vocabulary drastically. Accordingly, the decoder output vocabulary only needs to contain the primitive actions and a handful of masks, instead of the entire KB. Also, we carefully craft a set of primitive actions that are necessary for the complex question answering task and simplify the query form to reduce the search space. Our primitive actions free the model from the need for maintaining expensive and sophisticated memory

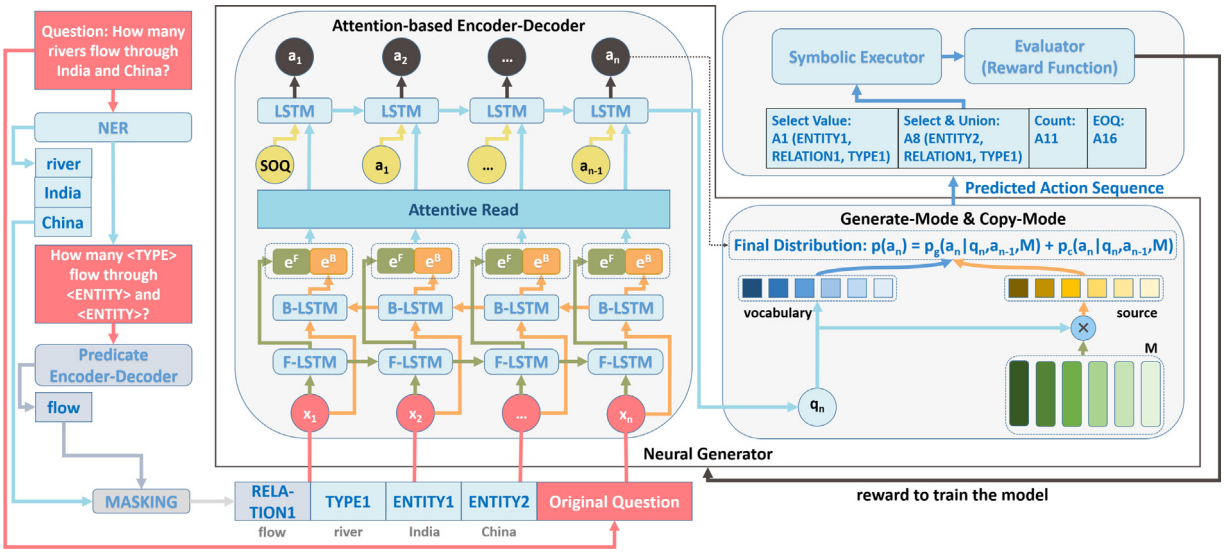

Fig. 2. Overview of the proposed NS-CQA model.

modules. On the contrary, the model maintains the intermediate results in a simple key–value dictionary. The model can directly compute the final answer to a question by executing a correct sequence of primitive actions.

Different from previous neural-symbolic models for the complex question answering task, our NS-CQA framework is designed to **augment the sparse extrinsic reward** by a dense intrinsic reward and exploit trials efficiently.

Instead of using the manually defined high-level constraints as in CIPITR, we employ a random search algorithm to find a few pseudo-gold annotations that lead to correct answers. The pseudo-gold annotations are used as supervision signals to pre-train the model, which could guide the model to filter out infeasible action sequences. Consequently, in the initial training stage, we mitigate the cold start issue by using the pseudo-gold annotations as demonstration data to pre-train our model.

Moreover, instead of recording one trial for each question, we maintain a memory buffer to store the promising trials, i.e., the action sequences that lead to the correct answer or gain high rewards. In our work, on the one hand, we aim to converge a behavior policy to a target optimal policy. Thus we need to measure how similar/important the generated trials are to trials that the target policy may have made. We design a reward bonus, **proximity**, to favor these "similar/important" trials. On the other hand, to avoid overfitting the spurious trials, we also encourage the policy to explore in undiscovered search space, i.e., the space that is beyond the imitation of the pseudo-gold annotations. Therefore, we design another reward bonus, **novelty**, to the generated action sequences that are "different" from the trials stored in the memory buffer.

To adaptively control the exploration–exploitation trade-off, we employ a *curriculum learning* [15] scheme to dynamically change the influence of the two reward bonuses, namely the proximity and the novelty. Particularly, given a question, we define the proximity for the predicted trial as the highest similarity between the predicted trial and all the trials in the memory buffer. Novelty is also defined through similarity. We consider a trial is novel if it is not similar to all the trials in the memory buffer. At the initial stage, since the policy is sub-optimal and the trials generated by the policy are generally infeasible, we prefer more novelty for exploration. We gradually increase the proportion of proximity and reduce the influence of novelty during the later training epochs. At the final stage of the training, the proportion of proximity in the reward bonus will rise to 100%. Such a delicately designed curriculum method can significantly improve the learning quality and efficiency [16].

Besides, to alleviate the sparse reward problem, an adaptive reward function (ARF) is proposed. ARF encourages the model with partial reward and adapts the reward computing to different question types. We sum up the reward bonus with the adaptive reward to make our RL model learn from the combined reward. With this modification, we reshape the sparse rewards into dense rewards and enable any failed experience to have a nonnegative reward, thus alleviate the sparse reward problem.

Furthermore, by incorporating a memory buffer, which maintains off-policy samples, into the policy gradient framework, we **improve the sample efficiency** of the REINFORCE algorithm in our work [11]. We encourage the model to generate trials that are similar to the trials in the memory buffer by using the proximity bonus. Therefore the high-reward trials could be re-sampled frequently to avoid being forgotten in the training process. Since a group of question shares the same pattern, a sequence of the same actions could solve such questions. Once we improve sample efficiency, we reduce the trials needed for training. Therefore, using a minimal subset of training samples, our model can produce competitive results.

Overall, the main contributions in this paper can be summarized as follows.

1. A neural-symbolic approach that is augmented by memory buffer and is designed for complex question answering. In our method, for each question, we resort to the previous promising trials stored in the memory to assist our model in replaying and generating feasible action sequences.
2. A curriculum-learning scheme that adaptively combines novelty and proximity to balance the exploration-exploitation trade-off for the model. We treat the combination of the novelty and proximity as a bonus to the reward, therefore alleviate the sparse reward and data inefficiency problems.
3. Several simple yet effective techniques are proposed to reduce search space, improve model generalizability, and accelerate convergence. In our work, the masking method and the copy mechanism are incorporated in Seq2Seq learning to avoid searching over a large action space. Also, a set of primitive actions is carefully designed to solve complex questions by executing actions sequentially, avoiding the maintenance of complex memory modules.

Our experiments on a large complex question answering dataset [6] and a relatively smaller multi-hop dataset WebQuestionsSP [17] show that NS-CQA outperforms all the recent, state-of-the-art models. Moreover, NS-CQA performs well on the questions with higher complexity, demonstrating the effectiveness of our method.

The rest of this paper is organized as follows. Related works are introduced in Section 2. Our NS-CQA framework is described in Section 3. Section 4 describes the experiments and evaluation results. In Section 5, we perform some qualitative analysis, showing positive examples and typical errors. We conclude our work in Section 6.

2. Related work

The NS-CQA model is inspired by two lines of work: semantic parsing and neural-symbolic systems. Semantic parsing mainly focuses on reformulating natural language questions into logic forms, which are then executed on knowledge bases (KBs) to compute answers [18–20]. More recent approaches employ sophisticated deep learning models to search entities and predicates that are most relevant to the question [21–23].

Many of these works tackle simple one-hop questions that are answerable by a single triple. Others address the multi-hop task, in which answers are entities that can be retrieved by a path of connected triples. In both cases, a model only retrieves entities as answers in a fixed search space, i.e., the KB. Our model addresses a more challenging problem, where answers come from a much larger search space, involving not only entities in the KB but also their logical combinations and aggregations (numbers).

The complex KBQA dataset, such as CQA [6], requires the execution of discrete actions rather than merely searching for entities in KB. Memory network [4,24–26] is used to store facts in KB and makes it possible to solve complex questions. Luo et al. [27] encodes complex questions into vectors to represent the semantics and structure of the input sentence contemporaneously, by which the similarity between the question and the graph could be computed. Dialog-to-Action (D2A) [28] incorporates dialog memory management in generating logical forms that would be executed on a large KBs to answer complex questions. It labels all training samples with pseudo-gold actions and trains the model by imitation learning. It is worth noting that D2A aims to answer *context-dependent* questions, where each question is part of a multiple-round conversation. On the other hand, in this paper, we consider answering the single-turn questions. Therefore we do not include D2A as a baseline method in the evaluation as it is not directly comparable to our problem setup. The semantic parsing approaches mentioned so far all require the annotation of the entire training dataset to learn the models. Different from these approaches, our model can be learned from a small number of pseudo-gold annotations only.

Some recent studies in KBQA focus on semi-supervised learning, in which models are trained solely on denotations, e.g., the execution results of queries, the state of the final time step, etc. In [1], a semantic parser is trained by learning from question–answer pairs rather than annotated logical forms to query the knowledge graph. Furthermore, the parser could be trained without manual annotations or question–answer pairs by treating denotations of natural language questions and related KB queries as weak-supervision [29]. Similarly, queries automatically generated from knowledge graph triples and paraphrased questions without answers are used in weak-supervision to train subgraph embedding models [30]. To fully understand the intention of questions, STAGG [5] is proposed to generate a staged query graph and directly map the graph into λ-calculus.

Neural-symbolic models integrate neural networks with logic-based symbolic executors to conduct non-differentiable computations. Neural Turing Machines (NTMs) [31] pioneer the neural-symbolic methods, in which REINFORCE is employed to train the model in the RL Neural Turing Machines (RL-NTMs) [32]. When answering natural language questions on relational tables, some approaches [33,34] predict discrete symbolic operations by neural networks and obtain answers by executing them. Guu et al. [35] marry RL and maximum marginal likelihood (MML) to avert the spurious problems. The neural-symbolic visual question answering (NS-VQA) system [36] combines deep representation learning and symbolic program execution to solve visual question answering problems over a synthetic image dataset.

Most relevant to this work are two state-of-the-art techniques on complex KBQA: Neural Symbolic Machines (NSM) [8] and CIPITR [37]. NSM deals with multi-hop questions in the WebQuestionsSP dataset [17] with two components: the programmer and the computer. By employing an EM-like mechanism, NSM iteratively finds the pseudo-gold trials for the training questions. NSM then assigns the pseudo-gold trials with a deterministic probability, therefore, to anchor the model to the high-reward trials. In a similar vein, CIPITR translates a natural-language complex question into a multi-step executable program using the Neural Program Induction (NPI) technique. CIPITR does not require gold annotations and can learn from auxiliary rewards, KB schema, and inferred answer types.

Our neural-symbolic model is different from them in that we augment our RL-based model with a memory buffer to record the successful trials. With the help of the memory buffer, we could compute the extra reward bonus to encourage the model to generate new trials, imitate successful trials, and reshape the sparse reward to provide dense feedback. As can be seen in Table 3, our NS-CQA model outperforms all the baseline models, and the performance difference is prominent on the more complex categories of questions. Also, in Table 4, we can find that NS-CQA is better than other baseline models. The superiority of NS-CQA in both the datasets verified the effectiveness and generalization ability of the model.

3. NS-CQA: A complex question answering approach

In our work, the complex question answering problem is regarded as a semantic parsing task: given a complex question q consisting of tokens $(w_1, \ldots, w_m)$, the model generates a primitive action sequence $(a_1, \ldots, a_q)$, and execute the sequence on the KB $\mathcal{K}$ to yield the answer a.

This section outlines our NS-CQA approach to the complex question answering problem. We first describe the set of primitive actions in Section 3.1. Given a complex question, the *semantic parser* (Section 3.2) recognizes KB artifacts that are relevant to the question. By combining the question and the output of the parser, the *neural generator* (Section 3.3) transforms the query into a sequence of primitive actions.

The *symbolic executor* (Section 3.4) executes the actions on the KB to obtain an answer. Overall, we employ *RL* to directly optimize the generator through a policy gradient on the answer predicted by the executor (Section 3.5). The high-level architecture of our model is depicted in Fig. 2.

3.1. Primitive actions

We propose a set of primitive actions based on the subset of SPARQL queries that are necessary for the current complex question answering task and simplify the query form to reduce the search space. Our actions are designed to be simple, dispensing with SPARQL features, including namespaces, etc. It also does

Table 1
Demonstration of how intermediate result evolves when executing actions sequentially.

	Actions	**Action1:** *SelectAll (country, flow, river)*	**Action2:** *ArgMax*	**Action3:** *EOQ*
Question1: Which country has **maximum number** of rivers?	Retrieved Key–Value Pairs	{China:{Indus, Satluj}} {India:{Indus, Satluj, Godavari}} {Russia:{Volga, Moskva, Neva, Ob}} {USA:{Mississippi, Colorado, Rio Grande}}	/	/
	Dictionary	{China:{Indus, Satluj}} {India:{Indus, Satluj, Godavari}} {Russia:{Volga, Moskva, Neva,Ob}} {USA:{Mississippi, Colorado, Rio Grande}}	{Russia:{Volga, Moskva, Neva, Ob}}	{Russia:{Volga, Moskva, Neva, Ob}}
	Actions	**Action1:** *Select (India, flow, river)*	**Action2:** *Diff (China, flow, river)*	**Action3:** *EOQ*
Question2: What rivers flow in India **but not** China?	Retrieved Key–Value Pairs	{India:{Indus, Satluj, Godavari}}	{China:{Indus, Satluj}}	/
	Dictionary	{India:{Indus, Satluj, Godavari}}	{India:{Godavari}}	{India:{Godavari}}

not support certain SPARQL features, including OPTIONAL, FILTER, etc. Since our actions belong to a subset of SPARQL's operators, the complexity of our actions follows that of SPARQL. The main contributions of this paper relate to the neural generator of these programs. Thus we leave the study of operator complexity to future research.

Unlike NSM [8], which introduces the variables to save the intermediate results, we employ a **key–value memory** to maintain the intermediate result. NSM adds a new intermediate variable into the decoder vocabulary after an action is executed, thus enables the decoder to generate the new variable in later decoding steps. Consequently, NSM dynamically increases the size of the decoder vocabulary in the decoding process. On the contrary, with the help of the memory, our model does not need to refer to previous intermediate variables, thus fix the size of the decoder vocabulary to simplify the model.

We use two components, i.e., an operator and a list of variables, to compose the primitive actions. After analyzing different types of complex problems, we design 17 operators in this work, which are described in Table 2. Besides, the key–value dictionary $\mathcal{D}$ is designed to store intermediate results. Keys in $\mathcal{D}$ refer to entities present in the question or specific special symbol, and the values are the obtained elements related to it. Before the execution of the whole action sequence, $\mathcal{D}$ is initialized as empty. When executing an action, the model will generate an intermediate result based on content in $\mathcal{D}$, which is the result derived from the last action. Then the generated intermediate result will be further stored in $\mathcal{D}$ to update it. The contents of updated $\mathcal{D}$ are then preserved for later use.

For instance, when dealing with the question "Which country has *maximum number* of rivers?", the desired output actions should be "*SelectAll(country, flow, river), ArgMax, EOQ*". The first action has an operator *SelectAll*, a relation variable *flow* and two type variables, i.e., *country* and *river*, while no entity variable is found in this action. Note that the second action only has one operator *ArgMax*, and so does the third action *EOQ*. By performing the first action, we retrieve KB to find all entities belong to type 'country' as keys. Meanwhile, we set the river-type entities linked with country-type entities by relation 'flow' as values. Like what is demonstrated in Table 1 (the retrieved results presented in the table are not consistent with the actual KB while are only for demonstration), one country-type entity is *USA* where the linked river-type objects are *{Mississippi, Colorado, Rio Grande}*. The retrieved key–value pairs (the key is one country-type entity and value is a set of river-type entities) are then stored in dictionary $\mathcal{D}$ as the intermediate result of this action. After that, the second action is executed to find the key whose mapped value has most elements. In could be found in Table 1 *Russia* has most elements thus *Russia:{Volga, Moskva, Neva, Ob}* is then kept in $\mathcal{D}$ and other key–value pairs are removed. Then we update $\mathcal{D}$ and view this key–value pair as the intermediate result of the second action. When encountered with *EOQ*, the model outputs the final result in $\mathcal{D}$.

To simplify the action sequence, we design particular actions (GreaterThan, LessThan, Inter, Union, and Diff) as relatively '**high-level**' actions that need multiple set operations to perform. Though such design will enhance the difficulty of symbolic executor, on the other hand, it could reduce the complexity of the neural generator. Since the bottleneck of our model lies in the difficulty of training generator, we make a compromise between executor and generator.

Take the question "What rivers flow in India *but not* China?" as an example. The reference action sequence should be "*Select(India, flow, river), Diff(China, flow, river), EOQ*". After executing the first action, a set of rivers flowing in India is stored in $\mathcal{D}$ as the value of the key *India*. As showed in Table 1, such key–value pair is *India:{Indus, Satluj, Godavari}*. When executing the second action "*Diff(China, flow, river)*", all river entities linked to *China* by relation *flow* are first retrieved from the KB. Then based on the key–value pair stored in $\mathcal{D}$, the entities indexed to India but not China will be kept as the updated value of the key *India*. In this case is *India:{Godavari}*. Then the key–value pair in $\mathcal{D}$ is updated accordingly. Upon encountering the action *EOQ*, the key–value pair stored in $\mathcal{D}$ is returned as the final answer to this question.

As described above, the model executes all the actions in sequence. With the help of a key–value dictionary $\mathcal{D}$, the intermediate result of the current action is recorded and preserved for later use. The model could perform the following actions based on the result stored in $\mathcal{D}$, and the new result would further update $\mathcal{D}$. Therefore, the design of $\mathcal{D}$ makes executing actions sequentially possible.

3.2. Semantic parser

Given a natural language question, the parser first recognizes entity mentions (for example *India*) and class mentions (for example *river*) [38]. A **Bidirectional-LSTM-CRF** model is employed to label the entity/type mentions [39]. The parser then links them with the corresponding entities and types in KB. At first, the parser tries to retrieve the entity/type candidates related to mentions by computing the literal similarities. Besides, the description of the candidates and the question are embedded into vectors to get semantic similarity. The literal and semantic similarities are thus integrated to rank the entity/type candidates, while the ones have the highest score is selected as linked entities/types. Subsequently, the entity/class mentions in the query are replaced

Table 2
The set of primitive actions. $\mathcal{K}$ represents the knowledge base, $e, e_1, e_2, \ldots$ represent entities, r represents a relation, t represents a type, and $\mathcal{D}$ represents a key–value dictionary which stores intermediate results.

ID	Action	Retrieved Key–Value Pairs	Output
A1	$Select(e, r, t)$	$\{e_2 \mid e_2 \in t, (e, r, e_2) \in \mathcal{K}\}$	$\mathcal{D} = \mathcal{D} \cup \{e : \{e_2\}\}$
A2	$SelectAll(et, r, t)$	$\{e_2 \mid e_1 \in et, e_2 \in t, (e_1, r, e_2) \in \mathcal{K}\}$	$\mathcal{D} = \mathcal{D} \cup \{e_1 : \{e_2\}\}$
A3	$Bool(e)$	$value = 1$ *if* $e \in \mathcal{D}$; *otherwise* $value = 0$	$\mathcal{D} = \{bool : value\}$
A4	$ArgMin$	$\{e_1 \mid e_1 \in \mathcal{D}, \exists(e_1 : \{e_2\}) \in \mathcal{D}, \forall e' : (e' : \{e'_2\}) \in \mathcal{D}, \lvert\{e_2\}\rvert \leq \lvert\{e'_2\}\rvert\}$	$\mathcal{D} = \{e_1 : \{e_2\}\}$
A5	$ArgMax$	$\{e_1 \mid e_1 \in \mathcal{D}, \exists(e_1 : \{e_2\}) \in \mathcal{D}, \forall e' : (e' : \{e'_2\}) \in \mathcal{D}, \lvert\{e_2\}\rvert \geq \lvert\{e'_2\}\rvert\}$	$\mathcal{D} = \{e_1 : \{e_2\}\}$
A6	$GreaterThan(e)$	$\{e_1 \mid e_1 \in \mathcal{D}, \exists(e_1 : \{e_2\}) \in \mathcal{D}, \exists e'_2 : (e : \{e'_2\}) \in \mathcal{D}, \lvert\{e_2\}\rvert \geq \lvert\{e'_2\}\rvert\}$	$\mathcal{D} = \{e_1 : \{e_2\}\}$
A7	$LessThan(e)$	$\{e_1 \mid e_1 \in \mathcal{D}, \exists(e_1 : \{e_2\}) \in \mathcal{D}, \exists e'_2 : (e : \{e'_2\}) \in \mathcal{D}, \lvert\{e_2\}\rvert \leq \lvert\{e'_2\}\rvert\}$	$\mathcal{D} = \{e_1 : \{e_2\}\}$
A8	$Inter(e, r, t)$	$\{e_2 \mid e_2 \in t, (e, r, e_2) \in \mathcal{K}\}$	$\mathcal{D} = \mathcal{D} \cap \{e : \{e_2\}\}$
A9	$Union(e, r, t)$	$\{e_2 \mid e_2 \in t, (e, r, e_2) \in \mathcal{K}\}$	$\mathcal{D} = \mathcal{D} \cup \{e : \{e_2\}\}$
A10	$Diff(e, r, t)$	$\{e_2 \mid e_2 \in t, (e, r, e_2) \in \mathcal{K}\}$	$\mathcal{D} = \mathcal{D} - \{e : \{e_2\}\}$
A11	$Count$	$Card(\mathcal{D}) = \lvert\{e \mid e \in \mathcal{D}, \exists(e : \{e_2\}) \in \mathcal{D}\}\rvert$	$\mathcal{D} = \{num : Card(\mathcal{D})\}$
A12	$AtLeast(n)$	$\{e \mid e \in \mathcal{D}, \exists(e : \{e_2\}) \in \mathcal{D}, \lvert\{e_2\}\rvert \geq n\}$	$\mathcal{D} = \{e : \{e_2\}\}$
A13	$AtMost(n)$	$\{e \mid e \in \mathcal{D}, \exists(e : \{e_2\}) \in \mathcal{D}, \lvert\{e_2\}\rvert \leq n\}$	$\mathcal{D} = \{e : \{e_2\}\}$
A14	$EqualsTo(n)$	$\{e \mid e \in \mathcal{D}, \exists(e : \{e_2\}) \in \mathcal{D}, \lvert\{e_2\}\rvert = n\}$	$\mathcal{D} = \{e : \{e_2\}\}$
A15	$GetKeys$	$\{e \mid e \in \mathcal{D}, \exists(e : \{e_2\}) \in \mathcal{D}\}$	$\mathcal{D} = \{key : \{e\}\}$
A16	$Almost(n)$	$\{e \mid e \in \mathcal{D}, \exists(e : \{e_2\}) \in \mathcal{D}, \lvert\lvert\{e_2\}\rvert - n\rvert \leqslant \alpha\}$ where α is predefined	$\mathcal{D} = \{e : \{e_2\}\}$
A17	EOQ	*End of sequence*	$\mathcal{D}$

with wild-card characters to generate patterns. We employ a **convolutional Seq2Seq model** [40] to transform the generated patterns to corresponding relations.

3.3. Neural generator

Our generator is an attention-based Seq2Seq model augmented with the copy and masking mechanisms. Given a question with tokens $(w_1, \ldots, w_m)$, the generator predicts tokens $(a_1, \ldots, a_q)$. The input of the model is the original complex question concatenated with KB artifacts generated by the semantic parser, and the output is the tokens of a sequence of actions. In our work, the output at each time step is a single token which is used to compose actions with adjacent output tokens.

For a vanilla Seq2Seq model, all the KB artifacts corresponding to all questions will need to be collected in advance to make the vocabulary large enough to cover all questions. Let N represent the maximum number of actions in sequences. Since we have designed 17 different operators in our work, and each of operators can take up to three arguments (see Table 2 for details), in the worst case, the vocabulary size of the decoder is $O(17^N * |\mathcal{E}|^{2N} * |\mathcal{P}|^N)$, where $|\mathcal{E}|$ and $|\mathcal{P}|$ denote the number of entities (including types) and predicates in the KB $\mathcal{K}$ respectively. Given a large KB such as Freebase, $|\mathcal{E}|$ and $|\mathcal{P}|$ can be very large. Such a vocabulary size would be prohibitively large for the decoder, making it highly unlikely to generate the correct token, thus negatively affecting the rate of convergence.

By incorporating the masking mechanism, the names of KB artifacts used for compose actions are replaced with *masks* such as `<ENTITY1>`, `<TYPE1>` and `<PREDICATE1>`. Thus, an action sequence consisted of N actions will be masked into the following form: $A^{(1)}(\langle \mathrm{E}_1 \rangle, \langle \mathrm{P}_1 \rangle, \langle \mathrm{E}_2 \rangle), \ldots, A^{(N)}(\langle \mathrm{E}_{2N-1} \rangle, \langle \mathrm{P}_N \rangle, \langle \mathrm{E}_{2N} \rangle)$. For instance, the action sequence '*Select(India, flow, river), Diff(China, flow, river), EOQ*' is used to solve the problem 'What rivers flow in India but not China?'. After masking, the real names of artifacts in actions are substituted with *masks*, and the action sequence is changed into '*Select(ENTITY1, PREDICATE1, TYPE1), Diff(ENTITY2, PREDICATE1, TYPE1), EOQ*'. The mappings between the actual names and *masks* are recorded in our model, which will be later used to recover the exact names of KB artifacts in actions when being executed. Given the maximum number of actions N, with the masking mechanism, the decoder vocabulary size is reduced to $O(17^N * (2N)^{2N} * N^N)$, where $(2N) \ll |\mathcal{E}|$ and $N \ll |\mathcal{P}|$. As action sequences are typically not long (i.e., $N \leq 5$ in our observation), this represents orders of magnitude reduction in vocabulary size.

Also, we could alleviate the Out Of Vocabulary (OOV) problem with the help of the masking mechanism. OOV words refer to unknown KB artifacts that appear in the testing questions but not included in the output vocabulary and would make the generated action incomplete. When facing a question with unseen KB artifacts, the model is not able to predict such objects since they are out of the output vocabulary. However, when employing the masking mechanism, all the KB artifacts are translated into masks, thus enabling the model to select masks from relatively fixed output vocabulary. Like the above example presented, the KB artifacts 'India' and 'China' are replaced with the mask 'ENTITY1' and 'ENTITY2'. Thus our model only needs to generate the masked tokens instead of the real names of the KB artifacts, which will mitigate the OOV problem. In consequence, the model could form the actions more precisely.

To further decrease search space, the copy mechanism is also incorporated. The copy mechanism *replicates* all masked symbols in the input sequence to form the output, instead of letting the decoder generate them from the decoder vocabulary. As a result, the decoder only needs to generate primitive actions, further reducing vocabulary size to $O(17^N)$.

The benefits of our design are twofold. (1) The much-reduced vocabulary makes convergence faster, as the generator is only concerned about generating correct primitive actions, but not names of artifacts from the KB. (2) Solve questions with unforeseen KB artifacts by directly masking and copying them from the input question when generating an action sequence.

Encoder. The encoder is a bidirectional LSTM that takes a question of variable length as input and generates an encoder vector $\boldsymbol{e}_i$ at each time step i.

$$\boldsymbol{e}_i, \boldsymbol{h}_i = LSTM(\phi_E(x_i), \boldsymbol{h}_{i-1}). \tag{1}$$

Here ϕ_E is word embedding of token E. $(\boldsymbol{e}_i, \boldsymbol{h}_i)$ is the output and hidden vector of the ith time step when encoding. The dimension of $\boldsymbol{e}_i$ and $\boldsymbol{h}_i$ are set as the same in this work. $\boldsymbol{e}_i$ is the concatenation of the forward $(\boldsymbol{e}_i^F)$ and backward $(\boldsymbol{e}_i^B)$ output vector and $\boldsymbol{h}_i$ is

the encoder hidden vector. The output vectors $(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_T)$ is regarded as a short-term memory $\boldsymbol{M}$, which is saved to use in copy mode.

Decoder. Our decoder of NS-CQA predicts output tokens following a mixed probability of two models, namely **generate-mode** and **copy-mode**, where the former generates tokens from the fixed output vocabulary and the latter copies words from the input tokens. Furthermore, when updating the hidden state at time step t, in addition to the word embedding of predicted word at time $t-1$, the location-based attention information is also utilized.

By incorporating the copy mechanism, the generated actions might be chosen from the output vocabulary or input tokens. We assume a fixed output vocabulary $\mathcal{V}_{output} = \{v_1, \ldots, v_N\}$, where $\mathcal{V}_{output}$ contains operators and related arguments in action sequence. In addition to that, all the unique words from input tokens $x = \{x_1, \ldots, x_T\}$ constitute another set $\mathcal{X}$ whereby some OOV words could be 'copied' when such words are contained in $\mathcal{X}$ but not in $\mathcal{V}_{output}$. Therefore, the vocabulary unique to input tokens x is: $\mathcal{V}_x = \mathcal{V}_{output} \cup \mathcal{X}$.

The **generate-mode** predicts output token a_t from the output vocabulary $\mathcal{V}_{output}$. Like traditional Seq2Seq model, the decoder is another LSTM model that generates a hidden vector $\boldsymbol{q}_t$ from the previous output token a_{t-1}. Previous step's hidden vector $\boldsymbol{q}_{t-1}$ is fed to an attention layer to obtain a context vector $\boldsymbol{c}_t$ as a weighted sum of the encoded states. Current step's $\boldsymbol{q}_t$ is generated via:

$$\boldsymbol{q}_t = LSTM(\boldsymbol{q}_{t-1}, [\phi_D(a_{t-1}), \boldsymbol{c}_t]) \tag{2}$$

Here ϕ_D is the word embedding of input token a_{t-1}. The dimension of $\boldsymbol{q}_t$ is set as d_q, and the attention weight matrix is trainable. The hidden vector $\boldsymbol{q}_t$ is used to compute the score of target word v_i in $\mathcal{V}_{output}$ as:

$$\psi_g(a_t = v_i) = \boldsymbol{v}_i^\top \boldsymbol{W}_o \boldsymbol{q}_t, \ \ v_i \in \mathcal{V}_{output} \tag{3}$$

where $\boldsymbol{W}_o$ is trainable matrix and $\boldsymbol{v}_i$ is the vector of word v_i.

In **copy-mode**, the score of "copying" word x_j from input tokens $\{x_1, \ldots, x_T\}$ is computed as:

$$\psi_c(a_t = x_j) = \sigma(\boldsymbol{e}_j \boldsymbol{W}_c \boldsymbol{q}_t), \ \ x_j \in \{x_1, \ldots, x_T\} \tag{4}$$

where $\boldsymbol{W}_c \in \mathbb{R}^{d_q \times d_q}$, σ is a non-linear activation function and, hidden encoder vectors $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_T\}$ in short-term memory $\boldsymbol{M}$ are used to map the input tokens $\{x_1, \ldots, x_T\}$ respectively.

Finally, given the hidden vector $\boldsymbol{q}_t$ at time t and short-term memory $\boldsymbol{M}$, the output token a_t is generated following a mixed probability as follows:

$$p(a_t | \boldsymbol{q}_t, a_{t-1}, \boldsymbol{M}) = p_{cg}(a_t | \boldsymbol{q}_t, a_{t-1}, \boldsymbol{M}) \tag{5}$$

where p_g and p_c indicate the generate-mode and copy-mode respectively. They are calculated as follows:

$$p_{cg} = \begin{cases} \frac{1}{Z} e^{\psi_g(a_t)}, & a_t \in \mathcal{V}_{output} \\ \frac{1}{Z} \Sigma_{j: x_j = a_t} e^{\psi_c(a_t)}, & a_t \in \mathcal{X} \end{cases} \tag{6}$$

where Z is the normalization term and is computed as: $Z = \Sigma_{v \in \mathcal{V}_{output}} e^{\psi_g(v)} + \Sigma_{x \in \mathcal{X}} e^{\psi_c(x)}$.

At time step t, word embedding of previous output token a_{t-1} and the location-based attention information are both employed to update the hidden state. a_{t-1} will be represented as $[\phi_D(a_{t-1}); \boldsymbol{r}_{q_{t-1}}]$, where $\phi_D(a_{t-1})$ is the word embedding of a_{t-1} and $\boldsymbol{r}_{q_{t-1}}$ is the weighted sum of hidden states $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_T\}$ in $\boldsymbol{M}$.

Vector $\boldsymbol{r}_{q_{t-1}}$ is calculated as:

$$\boldsymbol{r}_{q_{t-1}} = \Sigma_{\tau=1}^{T} \rho_{t\tau} \boldsymbol{e}_\tau \tag{7}$$

$$\rho_{t\tau} = \begin{cases} \frac{1}{K} p_{cg}(x_\tau | \boldsymbol{q}_{t-1}, a_{t-1}, \boldsymbol{M}), & x_\tau = a_{t-1} \\ 0, & otherwise \end{cases} \tag{8}$$

where K is the normalization term which is $\Sigma_{\tau': x_{\tau'} = a_{t-1}} p_{cg}(x_{\tau'} | \boldsymbol{q}_{t-1}, a_{t-1}, \boldsymbol{M})$, considering there might be input tokens located at different positions which equal to a_{t-1}. $\rho_{t\tau}$ is viewed as location-based attention in our work.

3.4. Symbolic executor

A symbolic executor is implemented as a collection of deterministic, generic functional modules to execute the primitive actions, which have a one-to-one correspondence with the functional modules. The symbolic executor first analyzes the output tokens produced by neural generator, and would assemble the actions one by one. Given an action sequence that begins with the first action, the symbolic executor executes the actions in order, on the intermediate result of the previous one. As discussed in Section 3.1, this is only possible due to our carefully designed primitive actions. Otherwise, complex memory mechanisms would need to be incorporated to maintain intermediate answers [8,28]. Upon encountering the action *EOQ*, the result from the last execution step will be returned as the final answer.

3.5. Training paradigm

As the symbolic executor executes non-differentiable operations against a KB, it is difficult to utilize end-to-end back-propagation to optimize the neural generator. Therefore, we adopt the following two-step procedure to train the generator. By using a breadth-first-search (BFS) algorithm, we generate pseudo-gold action sequences for a tiny subset of questions. In BFS, we assemble all the operators and KG artifacts found in question to form candidate action sequences in a brute-force way. We then execute the candidate action sequences to find the ones that yield the right answer and view them as pseudo-gold action sequences. Using these pairs of questions and action sequences, we pre-train the model by Teacher Forcing.

We then employ RL to fine-tune the generator on another set of question–answer pairs. The symbolic executor executes the predicted action sequence to output an answer and yield a reward for RL. The reward is the similarity between the output answer and the gold answer.

As shown in Algorithm 1, our method works as follows. The training starts with an empty memory buffer, and at every epoch, for each sample, the generated trials that gain high reward will be stored in the memory. For each question, we first use a search algorithm, i.e., greedy-decoding, to generate a trial. We execute the trial and compute a reward r_{greedy}, which is set as the reward threshold. Then we employ a beam search method to generate a set of candidate trials for the question and compute their rewards. At every epoch, we compare the generated trials with the trials in memory to determine the reward bonus, aka proximity and novelty, and further add the reward bonus to the adaptive reward. A candidate trial is added into the memory buffer if its reward is higher than r_{greedy}. We utilize the augmented reward to train the policy under the RL setting.

The memory buffer stores a limited set of trials for the training questions. Once the memory buffer is full, we substitute a random trial with the current new trial. This strategy enables the memory buffer to maintain relatively fresher trials, but would not always abandon the older ones.

The main components of our training paradigm, namely the RL method, the adaptive reward, and the curriculum reward bonus, are described in the rest of this section.

Algorithm 1: Training NS-CQA

Input: Training dataset Q_{train}, initial policy θ, memory buffer M, reward function $R(\cdot)$, learning rate η_1
Output: The learned policy θ^*

Randomly initialize θ
$M \leftarrow \emptyset$
while *not converged* **do**
  Sample batch of data $Q_{batch} \sim Q_{train}$
  $\mathcal{L} \leftarrow 0$
  for $q \in Q_{batch}$ **do**
    Get one trial $\boldsymbol{t_{greedy}}$ by greedy-decoding
    Compute adaptive reward r_{greedy}
    Compute cumulative reward $R(q, \boldsymbol{t_{greedy}})$
    Sample K trials: $\boldsymbol{t_k} \sim \pi(\boldsymbol{t}|q;\theta)$
    for *each trial* $\boldsymbol{t_k}$ **do**
      Compute adaptive reward $r_{\boldsymbol{t_k}}$
      Compute cumulative reward $R(q, \boldsymbol{t_k})$
      Update memory: add $\boldsymbol{t_k}$ to M if $r_{\boldsymbol{t_k}} > r_{greedy}$
    end
    $L = \frac{1}{K}\sum_{k=1}^{K}[R(q, \boldsymbol{t_k}) - R(q, \boldsymbol{t_{greedy}})]log(p_\theta(\boldsymbol{t_k}))$
    $\mathcal{L} \leftarrow \mathcal{L} + L$
  end
  Compute adapted parameters: $\theta \leftarrow \theta + \eta_1 \nabla_\theta \mathcal{L}$
end
Return The learned policy $\theta^* \leftarrow \theta$

Reinforcement learning. In this step, REINFORCE [41] is used to finetune the neural generator. Typically, the three notions mentioned in RL are action, state, and reward, respectively. In our scenario, at each step, action as a fundamental concept in RL is a token produced by a neural generator used to form an executable action sequence. What needs to be clarified is that when related to RL, the concept of actions refers to the tokens of a trial. Since the complex question answering environment is deterministic, we define the state as the question combined with the generated tokens so far. Meanwhile, the reward is the same as the notion in RL.

In our work, the state, action and reward at time step t are denoted as s_t, a_t and r_t respectively. Given a question q, the state of time step t is defined by q and the action sequence so far: $s_t = (q, a_{0:t-1})$, and the action tokens are generated by the generator. At the last step of decoding T, the entire sequence of actions is generated. The symbolic executor will then execute the action sequence to produce an output answer ans_o. Therefore the reward is computed only after the last step of decoding when ans_o is output. We design a Adaptive Reward Function (ARF), which is the adaptive comparison of the output answer ans_o and the gold answer ans_g. Furthermore, we employ a curriculum-guided Reward Bonus (CRB), which comprises proximity and novelty, to assign non-zero rewards for actions that do not yield correct answers. Specifically, the cumulative reward of an action sequence $a_{0:T}$ is the sum of CRB and ARF:

$$R(q, a_{0:T}) = CRB + ARF(ans_o, ans_g) \tag{9}$$

Then $R(q, a_{0:T})$ is sent back to update parameters of the neural generator through a REINFORCE objective as the supervision signal.

Adaptive reward. Though the search space is significantly reduced to $O(17^N)$ after the masking and copy mechanisms are incorporated, the length of action sequence, which is N, would be fairly long when solving a relatively more complex question. In that case, $O(17^N)$, the size of the search space, is still huge which makes it hard for the model to find correct action sequences when gold annotations are unavailable.

Moreover, since the reward used to train the model could only be obtained after a sequence of actions is executed, the execution of actions is regarded as a part of training. Suppose a large amount of candidate action sequences (normally more than 50) are generated, their execution would consume a large amount of time since it involves searching triples in KB, performing set operations and discrete reasoning. To reduce the training time, we limit our NS-CQA model to form only 5∼20 candidate action sequences with a smaller beam size. With the huge search space and small beam size, the sparsity of the reward becomes a problem. Of all the candidate action sequences that are predicted, very few of them could output correct answers and be positively rewarded while most of them do not produce any reward. Under such circumstances, without the notion of partial reward, the neural generator would suffer from high variance and instability. Therefore the generator would be inclined to be trapped in local optima and not generalize well on data never seen before.

Furthermore, different categories of questions entail different answer types. The reward function should be adaptive to the diverse answer types which could measure the degree of correctness of predicted answers more precisely. In other words, the reward function should encourage the generator to generate action sequences with the correct answer type while punishing the model if the predicted answer type is incorrect.

In the complex question answering scenario, the possible types of answers are integers, sets of entities and Boolean values. To measure the answer correctness more accurately and adaptively, and to compute partial reward to alleviate the sparse reward problem, we define our adaptive reward function *ARF*. *ARF* computes reward based on different answer types using the function *Sim* between the output answer ans_o and the gold answer ans_g.

$$Sim(ans_o, ans_g) = \begin{cases} 1 - \frac{|ans_g - ans_o|}{|ans_g + ans_o + \varepsilon|}, & \text{integer} \\ Edit(ans_g, ans_o), & \text{Boolean} \\ F1(ans_g, ans_o), & \text{set} \end{cases} \tag{10}$$

The edit-score is used to measure the accuracy of the output provided the answer type is Boolean, while the F1-score is used as a reward when answer is a set of entities. When the answer type is Boolean, the expected output is a list of Boolean value, for instance as what is presented in Table 6, the expected answer of the question "s Alda Pereira-Lemaitre a citizen of France and Emmelsbull-Horsbull?" is [True, False]. Regard each Boolean value as a single element in a list, the edit (Levenshtein) distance is used to compute the similarity between two lists, i.e., output answer list, and gold answer list. Thus the similarity is calculated as follows:

$$Edit(ans_g, ans_o) = 1 - \frac{Levenshtein(ans_g, ans_o)}{max(|ans_g|, |ans_o|)} \tag{11}$$

On the other hand, suppose the type of answer is a set of entities, F1-score is computed as:

$$F1(ans_g, ans_o) = 2 * \frac{precision * recall}{precision + recall} = 2 * \frac{\frac{|ans_g \cap ans_o|}{|ans_o|} * \frac{|ans_g \cap ans_o|}{|ans_g|}}{\frac{|ans_g \cap ans_o|}{|ans_o|} + \frac{|ans_g \cap ans_o|}{|ans_g|}} \tag{12}$$

If the answer type is incorrect or the action sequence is semantically invalid, reward is set as 0. On the other hand, if the answer type is the same as the gold answer, partial reward is granted. We then defined *ARF* as follows:

$$ARF(ans_o, ans_g) = R_{type} * (W_1 + W_2 * Sim(ans_o, ans_g)) \tag{13}$$

In our work, ε, W_1 and W_2 are predefined hyper-parameters and set as 0.001, 0.2 and 0.8 respectively. If the type of predicted answer is correct, R_{type} is set as 1, otherwise 0.

Curriculum-guided reward bonus. In many RL settings, the reward is positive only when a trial, i.e., a long sequence of actions generated by a policy, could yield the correct result. At the initial stage, since the policy is not yet fully-trained, out of all the generated trials, the rate of successful trials is rare. Thus, there is an insufficient number of collected successful trials for training the RL model, which causes the sparse reward problem.

Besides, the suboptimal policy will not explore the search space effectively since many sampled trials could be repeated. Moreover, the policy will forget the rare successful trials easily since the trials may not be re-sampled frequently. These factors often lead to the data inefficiency problem.

To solve the above problems, we design two reward bonuses to learn from failed trials. We introduce a memory buffer to record the high-reward trials for each training sample. We compare a generated trial with those stored in the memory buffer to see how similar it is to the recorded trials. Therefore we give *proximity bonus* to a generated trial even if it fails to yield the correct answer. By doing this, we could encourage the policy to re-sample the high-reward trials and accordingly reduce the frequency of generating infeasible trials. Also, we give *novelty bonus* to the generated trials that differ from the trials in the memory buffer. The novelty bonus encourages the policy to generate different trials, thus avoid being trapped by spurious ones. Once the reward is augment with the above two bonuses, the corresponding failed experience is assigned with a nonnegative reward and can contribute to learning the policy.

To balance proximity and novelty, we employ a curriculum-learning method to regulate their trade-off dynamically. In the earlier epochs, higher novelty can help the policy to explore unseen areas and generate more diverse trials. However, in the later epochs, such novelty will bring more noise (often as spurious trials) into training and distract the policy. At the later stage of training, the policy has gained sufficient knowledge about the tasks and is able to generate high-reward, promising trials. Therefore, proximity becomes more critical since it will encourage the policy to proceed towards the correct trials and to focus on learning how to generate promising trials.

Given a question q, suppose the high-reward trials $t_1^q, \ldots, t_m^q$ have already been stored in the memory buffer M. For one generated trial t, we compute the reward bonus CRB as:

$$CRB = \alpha(\lambda F_{prox}(t, M) + (1 - \lambda)F_{novel}(t, M)), \tag{14}$$

where $\alpha \in [0, 1]$ is the weight of the reward bonus and dependent on the scale of the task rewards. The term F_{prox} reflects the proximity of the trial t to the recorded trials in memory M while F_{novel} measures the novelty of t. The value of λ controls their relative proportion, which is adjusted by the curriculum learning method.

We compute the similarity between the generated trial t with one trial t_i^q in M by edit distance, which is:

$$s_i = Edit(t, t_i^q) \tag{15}$$

Thus we define the proximity F_{prox} as the highest similarity between the trial t and all the trials $t_1^q, \ldots, t_m^q$ in the memory buffer, which is:

$$F_{prox}(t, M) = \max_{1 \leqslant i \leqslant m}(s_i) \tag{16}$$

The term novelty F_{novel} measures the diversity of the trial t from the trials $t_1^q, \ldots, t_m^q$. We assign a high novelty to a generated trial if it is different from the trials in M, thus we define the novelty as:

$$F_{novel}(t, M) = \beta - \frac{1}{m}\Sigma_{i=1}^{m} s_i, \tag{17}$$

where $\beta \in [0, 1]$ is used to measure the diversity and is dependent on the scale of the similarity.

We employ a curriculum learning scheme to adaptively change the weight λ in Formula (14). We start from learning to generate novel trials with large diversity, and gradually focus on re-sampling the trials which have high proximity to the desired successful trials stored in the memory buffer. This method is achieved by progressively increasing the weight λ, which is exponentially increased λ with the training epochs:

$$\lambda = min\{1, (1 + \eta)^{\gamma}\lambda_0\}, \tag{18}$$

where $\eta \in [0, 1]$ is the learning pace which controls the curriculum learning, γ represents the number of the epochs that have been trained, and λ_0 is the initial weight of λ.

In our work, α, β, η, and λ_0 are hyper-parameters which are defined as 0.1, 1.0, 0.08, and 0.1, respectively in our work.

REINFORCE. At each time step, the output token generated by agent is decided by a certain policy (which is the generator in our work), and the probability that one token a is chosen is computed as below, where θ denotes model parameters:

$$\pi_\theta(q, a) = P_\theta(a_t = a|q, a_{0:t-1}) \tag{19}$$

Thus, the probability of an entire action sequence $a_{0:T}$ is given by:

$$P_\theta(a_{0:T}|q) = \prod_{t=1}^{T} P_\theta(a_t|q, a_{0:t-1}) \tag{20}$$

In (9), we define the cumulative reward $R(q, a_{0:T})$. Our objective is to maximize the expected cumulative reward. Therefore we use the policy gradient method such as the REINFORCE algorithm to finetune the generator. The objective and gradient are:

$$\begin{aligned} J^{RL}(\theta) &= \sum_q \mathbb{E}_{P_\theta(a_{0:T}|q)}[R(q, a_{0:T})] \\ \nabla_\theta J^{RL}(\theta) &= \sum_q \sum_{a_{0:T}} P_\theta(a_{0:T}|q) \cdot [R(q, a_{0:T}) \\ &\quad - B(q)] \cdot \nabla_\theta log P_\theta(a_{0:T}|q) \end{aligned} \tag{21}$$

$B(q) = R(q, \hat{a}_{0:T})$ is a baseline that reduces the variance of the gradient estimation without introducing bias. In our work, the baseline is set as what is used in the self-critical sequence training (SCST) [42]. Also, Monte Carlo integration is employed to approximate the expectation over all possible trials in the policy gradient method [41]. The training method is presented in Algorithm 1.

4. Experiments

We evaluated our model NS-CQA on a large-scale complex question answering dataset (CQA) [6], and a challenging multi-hop question answering dataset WebQuestionsSP [17].

The CQA dataset is generated from the facts stored in Wikidata [12], consisting of 944 K QA pairs for training and 100 K/156 K QA pairs for validation and test, respectively. The CQA dataset is characterized by the challenging nature of the questions. To answer them, discrete aggregation operators such as set union, intersection, min, max, counting, etc. are required (see Table 2 for more details). The CQA questions are organized into seven categories, as shown in Table 3. Some of these categories (e.g., Simple Question) have entities as answers, while others have numbers (e.g., Quantitative (Count)) or Boolean values (e.g., Verification (Boolean)) as answers. We used 'accuracy' as the evaluation metric for categories whose answer type is 'Verification', 'Quantitative (Count)', and 'Comparative (Count)'; and 'F1 measure' for other types of questions. However, to simplify the presentation and

stay consistent with literature [7,9], we denote 'accuracy' as 'F1 measure' in Table 3. Hence, the model performance was evaluated on the F1 measure in this paper. Furthermore, we computed the micro F1 and macro F1 scores for all the models based on the F1-scores of the seven question categories.

In our analysis of the CQA dataset, we found that the seven categories of questions vary substantially in complexity. We found that 'Simple' is the simplest that only requires two actions to answer a question, whereas 'Logical Reasoning' is more difficult that requires three actions. Categories 'Verification', 'Quantitative Reasoning', and 'Comparative Reasoning' are the next in the order of difficulty, which need 3–4 actions to answer. The most difficult categories are 'Quantitative (Count)' and 'Comparative (Count)', needing 4–5 actions to yield an answer. Saha et al. [7] drew a similar conclusion through manual inspection of these seven question categories.

The WebQuestionsSP dataset collects multi-hop questions, i.e., the questions require a chain of KB triples to answer, via the Google Suggest API. In comparison to the CQA dataset, WebQuestionsSP can be considered easier as it only contains multi-hop questions, and the answers are (sets of) entities only. It consists of 3098 question–answer pairs for training and 1639 questions for testing. We utilized the same evaluation metrics employed by [6,8], the F-1 measure, to evaluate model performance on the testing questions.

4.1. Model description

Our model is evaluated against three baseline models: HRED+KVmem [6], NSM [8] and CIPITR [7]. We used the open-source code of HRED+KVmem and CIPITR to train the models and present the best result we obtained. As the code of NSM has not been made available, we re-implemented it and further incorporated the copy and masking techniques we proposed. HRED+KVmem does not use beam search, while CIPITR, NSM, and our model all do for predicting action sequences. When inferring the testing samples, we used the top beam [7], i.e., the predicted program with the highest probability in the beam, to yield the answer.

HRED+KVmem [6] is the baseline model proposed together with the CQA dataset [6], which combines a hierarchical encoder–decoder with a key–value memory network. The model first encodes the current sentence with context into a vector, whereby a memory network retrieves the most related memory. Then the retrieved memory is decoded to predict an answer from candidate words. HRED+KVmem was designed specifically for the CQA dataset, thus was not included in our experiments on WebQuestionsSP.

NSM [8] is an encoder–decoder based model which is trained by weak-supervision, i.e., the answers to the questions. NSM first employs an Expectation-Maximization-like (EM-like) method to find pseudo-gold programs that attain the best reward. It iteratively uses the current policy to find the best programs and then maximizes the probability of generating such programs to optimize the policy. Then NSM replays *one* pseudo-gold trial that yields the highest reward for each training sample when employing REINFORCE to train the policy. It assigns a deterministic probability to the best trial found so far to improve the training data efficiency. NSM was at first proposed to solve the problems in WebQuestionsSP, and we reimplemented it to also handle the CQA dataset.

As presented in 3.1, unlike NSM, we do not refer to the intermediate variables when generating the tokens of a trial. Therefore it is unnecessary to incorporate the key-variable memory, which is used to maintain and refer to intermediate program variables in our work. We thus removed the key-variable memory component in the seq2seq model in our reimplementation of NSM.

CIPITR [7] employs an NPI technique that does not require gold annotations. Instead, it relies on auxiliary awards, KB schema, and inferred answer types to train an NPI model. CIPITR transforms complex questions into neural programs and outputs the answer by executing them. It designs high-level constraints to guide the programmer to produce semantically plausible programs for a question. The auxiliary reward is designed to mitigate the extreme reward sparsity and further used to train the CIPITR model. CIPITR is designed to handle the KBQA problems proposed in both CQA and WebQuestionsSP.

4.2. Training

The NS-CQA model was implemented in PyTorch with the model parameters randomly initialized.[1] The Adam optimizer is applied to update gradients defined in Formula (21). We used the fixed GloVe [39] word vectors to represent each token in input sequences and set each unique, unseen word a same fixed random vector. We set a learning rate of 0.001, a mini-batch size of 32 samples to pre-train the Seq2Seq model with pseudo-gold annotations. On average, after about 70 epochs, the Seq2Seq model would converge. Then we trained the REINFORCE model with a learning rate of 1e-4 and a mini-batch size of 8 on the pre-trained Seq2Seq model until accuracy on the validation set converged (at around 30 epochs).

As solving the entity linking problem is beyond the scope of this work, we separately trained an entity/class/relation linker. When training the NS-CQA model, the predicted entity/class/relation annotations along with the pseudo-gold action sequence (which are generated by a BFS algorithm) were used. The entity/class/relation annotations predicted by the respective linker were used when conducting experiments on the test dataset.

Incorporating the copy and masking mechanisms, our full model took a total of at most 3700 min to train 100 epochs (70 epochs for the Seq2Seq model and 30 epochs for REINFORCE) till convergence. Most of the time was spent on RL training, which is over 3633 min. In constraints, when we tried to train CIPITR [7], the model required over 24 h to complete one epoch of training while the max number of epochs is also set as 30.

Training with annotations would make the model learn to search in a relatively more accurate space, thus converging faster. However, the limited availability of annotations remains a bottleneck for model training in many CQA tasks. On the other hand, training without annotations but with denotations solely makes model convergence harder.

We married the two ideas together: training with a small number of annotations and then the denotations. First, we automatically produced pseudo-gold annotations for a small set (e.g., less than 1% of the entire CQA training dataset) of questions. The pseudo-gold annotations were utilized to pre-train the model to constrain the search space. After that, the model was further trained with only denotations.

[1] To encourage reproductivity, we have released the source code at https://github.com/DevinJake/NS-CQA.

Table 3
Performance comparison (measured in F1) of the four methods on the CQA test set. Best results for each category is **bolded**, and second best is underlined.

Method	HRED+KVmem	CIPITR-All	CIPITR-Sep	NSM	Vanilla	PG	NS-CQA
Simple question	41.40%	41.62%	**94.89%**	88.33%	85.13%	84.25%	88.83%
Logical reasoning	37.56%	21.31%	**85.33%**	81.20%	70.46%	68.37%	81.23%
Quantitative reasoning	0.89%	5.65%	33.27%	41.89%	47.96%	56.06%	**56.28%**
Comparative reasoning	1.63%	1.67%	9.60%	64.06%	54.92%	**67.79%**	65.87%
Verification (Boolean)	27.28%	30.86%	61.39%	60.38%	75.53%	83.87%	**84.66%**
Quantitative (Count)	17.80%	37.23%	48.40%	61.84%	66.81%	75.69%	**76.96%**
Comparative (Count)	9.60%	0.36%	0.99%	39.00%	34.25%	43.00%	**43.25%**
Overall macro F1	19.45%	19.82%	47.70%	62.39%	62.15%	68.43%	**71.01%**
Overall micro F1	31.18%	31.52%	73.31%	76.01%	74.14%	76.56%	**80.80%**

4.3. Results on CQA

Table 3 summarizes the performance in F1 of the four models on the full test set of CQA.

It must be pointed out that **CIPITR** [7] separately trained *one single model for each of the seven question categories*. We denote the model learned in this way as **CIPITR-Sep**. In testing, CIPITR-Sep obtained test results of each category by employing the corresponding tuned model [7]. In practical use, when trying to solve a complex question more precisely, CIPITR-Sep has to first trigger a classifier to recognize the question category. Only after acquiring the question categories information could CIPITR-Sep know which model to select to answer the question. If the number of question categories is increased, CIPITR-Sep needs to train more models, which will impede the system from generalizing to unseen instances. Besides, CIPITR also trained *one single model over all categories of training examples* and used this single model to answer all questions. We denote this single model as **CIPITR-All**. Therefore, we separately present the performance of these two variations of CIPITR in Table 3. On the other hand, we tuned NS-CQA on all categories of questions with one set of model parameters. Our model is designed to adapt appropriately to various categories of questions with one model, thus only needs to be trained once.

We also compared our full model, NS-CQA, with several model variants to understand the effect of our techniques presented in this work. Specifically, **Vanilla** is an imitation-learning model that was trained with pseudo-gold annotations. **PG** denotes the RL model that was optimized by the Policy Gradient algorithm based on the pre-trained model Vanilla. **NS-CQA** means the RL model that is equipped with all the techniques proposed in this work, notably the memory buffer and the reward bonus.

In Table 3, several important observations can be made.

1. Over the entire test set, our full model NS-CQA achieves the best overall performance of 71.01% and 80.80% for macro and micro F1, respectively, outperforming all the baseline models. The performance advantage on macro F1 over the four baselines is especially pronounced, by 51.56, 51.19, 23.31, and 8.62 percentage points over HRED+KVmem, CIPITR-All, CIPITR-Sep, and NSM respectively. Also, NS-CQA improves over the micro F1 performance of HRED+KVmem, CIPITR-All, CIPITR-Sep, and NSM by 49.62%, 49.28%, 7.49%, and 4.79%. Moreover, our model achieves best or second-best in all the nine items being evaluated (the seven categories, plus overall macro F1, and overall micro F1). The improvement is mainly due to the techniques presented in this work. We introduce masking and copy mechanisms to reduce the search space effectively and carefully design a set of primitive actions to simplify the trials, therefore enable the model to efficiently find the optimal trials. We also augment the RL model with a memory buffer, whereby the model could circumvent the spurious challenge, and remember the high-reward trials to re-sample them.
2. Out of the seven categories, our full model NS-CQA achieves the best performance in four categories: Quantitative Reasoning, Verification (Boolean), Quantitative (Count), and Comparative (Count), and second best in the rest three. In the hardest categories, Quantitative (Count) and Comparative (Count), NS-CQA is substantially superior over the four baseline models, and outperforms our PG model. Since the length of the questions in the hardest categories is usually higher than in the other categories, it is always hard to find correct trials. Under such circumstances, the memory buffer could make the model search in unknown space while keeping the previous high-reward trials in mind, which makes the model easier to train. This is the main reason that NS-CQA performs the best in the hardest categories.
3. CIPITR-Sep achieves the best performance in two *easy* categories, including the largest type, Simple Question. For the harder categories, it performs poorly compared to our model. Also, CIPITR-All, the single model that is trained over all categories of questions, performs much worse in all the categories than CIPITR-Sep, which learns a different model separately for each question category. For CIPITR-Sep, the results reported for each category are obtained from the model explicitly tuned for that category. A possible reason for CIPITR-All's significant performance degradation is that the model tends to forget the previously appeared high-reward trials when many infeasible trials are generated. Besides, the imbalanced classes of questions also deteriorates the performance of the model. Different from CIPITR, our model is designed to remember the high-reward trials when training.
4. NSM and NS-CQA both produce competitive results. The copy mechanism, masking method, and our carefully-defined primitive actions presented in this work were used in both models when we implemented them. By comparing the overall macro and micro F-1 score, it could be observed that NSM performed the best in all the four baseline models. This helps to validate the effectiveness of our proposed techniques. However, NSM is worse than NS-CQA in all categories, especially in the harder ones. Since NSM records one promising trial for each question, it might be faced with the spurious problem. Different from NSM, we design a memory buffer for recording all successful trials to circumvent this problem. Also, NSM only considers the correctness of the predicted answers when measuring the reward, hence suffers from the sparse reward problem. Unlike NSM, our NS-CQA model augments the reward with proximity and novelty to mitigate this problem. These two factors make our model superior to the NSM model in all question categories.
5. Both of our model variants perform competitively. In Table 3, it can be seen that the PG model, which was equipped with RL, performed better than the Vanilla model in five

Table 4
Performance comparison (measured in F1) of the four methods on the WebQuestionsSP test set. Best results is **bolded**.

Method	F1 measure
CIPITR-All	43.88%
NSM	70.61%
PG	70.72%
NS-CQA	**72.04%**

categories, but did not perform well in the two easy categories, Simple and Logical Reasoning. We analyzed the degeneration and found that for these two types of questions, usually, each question has only one correct sequence of actions. When training with PG, some noisy spurious trials were introduced by beam search and thus degraded the model's performance. Our full model is better than the PG model in six of the seven categories and substantially improved performance in the Logical Reasoning category. We compared the trials generated by the full model and the PG model, and found that many noisy trials are removed with the help of the memory buffer. That is the main reason for the improvement in the Logical Reasoning category. However, we also found that the full model performed worse than PG in Comparative Reasoning, which will be further investigated in the future.

The above results demonstrate the effectiveness of our technique. It is worth noting that our model is trained on only 1% of the training samples, whereas the baseline models use the entire training set. Besides, our approach uses one model to solve all questions, while CIPITR-Sep trains seven separate models to solve the seven categories of questions. Thus our model is virtually compared with seven individually training models used in CIPITR-Sep. However, our model still achieves best performance overall as well as in five of the seven categories.

4.4. Results on WebQuestionsSP

Table 4 summarizes the performance in the F1 measure of the four models on the full test set of WebQuestionsSP.

Similar to the CQA dataset, CIPITR also divided questions into five categories, and then separately train one model for each category. However, since the category information is not provided in the WebQuestionsSP dataset, we did not classify the questions and trained one single model, CIPITR-All, for all the training samples by using its open-source code.

From Table 4, we can observe that NS-CQA can indeed learn the rules behind the multi-hop inference process directly from the distance supervision provided by the question–answer pairs. Without manually pre-defined constraints, our model could learn basic rules from the pseudo-gold annotations, and further complete the rules by employing RL.

NS-CQA performed the best in the four models and significantly outperformed the CIPITR-All. The main reason is that it is hard for CIPITR-All to learn one set of parameters that fits the different samples.

Also, by introducing the masking and copy mechanism, NSM could achieve a performance competitive to our models PG and NS-CQA. By employing memory buffer, our NS-CQA model can alleviate the sparse reward problem and avert being trapped by spurious trials, which makes the model more robust, therefore achieving the best performance.

Furthermore, NS-CQA achieves the best result on both the CQA and WebQuestionsSP datasets, which attests to the effectiveness and the generalizability of our method.

Table 5
Ablation study on the CQA test set, showing the macro F1 score drop by removing each main component, or by learning from a subset of the training set. The Vanilla model has macro F1 of 62.15% as shown in Table 3.

Feature	Macro F1
Vanilla	62.15%
Masking	−37.10%
Copy	−11.52%
Attention	−4.30%
1000-training	−5.78%
2000-training	−4.09%

4.5. Model analysis

To study how the different components influence the performance of our seq2seq model, i.e., Vanilla, we conduct an ablation experiment as follows. Each of the main components: attention, copy mechanism, and masking method, is removed one at a time from the full seq2seq model to study how its removal affects model performance. We also study the effect of smaller training samples on performance, by using 1 K and 2 K samples for training, instead of 10 K used in the full model.

Table 5 summarizes performance degradation on the CQA test set, where the Vanilla model achieves a macro F1 score of 62.15%. It can be seen that the removal of masking produces the largest drop in performance, of 37.10%. Masking method significantly decreases the search space by replacing all the entity, relation and type names with wildcard tokens. This result demonstrates that although a simple approach, masking proves to be valuable for the CQA task.

When the copy mechanism is removed, performance decreases by 11.52%. This is consistent with our expectation since masking has already considerably decreased the search space, the improvements that copy mechanism could makes is relatively limited. Lastly, when training on fewer samples labeled with pseudo-gold actions, the model under-fits.

When training on even smaller datasets, the performance degradation is not as severe as we expected. With as a training set as small as 1000, our model is able to generalize well, only suffering a 5.78% drop on a test set of 15.6 K. With a training set of 2000 samples, our model suffers a modest 4.09% drop in performance. This study further demonstrates the robustness and generalizability of our model.

4.6. Sample size analysis

In this subsection, we analyze the effect of training samples of different sizes on our PG module. Since the WebQuestionsSP consists of a limited number of questions, it is hard to conduct the sample size analysis on it. Instead, we trained our model by using different CQA subsets to make a comparison. Specifically, given the same pre-trained model, we train the NS-CQA model on 0.2%, 0.4%, 0.6%, 0.8%, 1.0%, and 1.2% of total 944 K training samples. Note that the evaluation results of the full model presented in Section 4.3 are obtained from 1.0% of training data and the entire test set (i.e., 156 K). For experiments described in this subsection, evaluation is performed on a subset of the full test set that is 10% of its size (i.e., 15.6 K). Training of the REINFORCE model is stopped at 30 epochs, which is when all models have been observed to converge.

We first study the effect on model performance. Fig. 3 plots the macro F1 values of the seven categories of questions as well as the overall performance. With the increase in training data size, a general upward trend in macro F1 values can be observed, with the category Simple Question being the exception. For the overall

test set, we can observe that the macro F1 value plateaus at 1.0% and does not increase when training data is expanded to 1.2%.

For Simple Question, the output actions are relatively the 'simplest'. In most cases, one 'Select' action is needed to solve a question. Consequently, with the help of methods to decrease search space and better use data, after pre-training, the model overfits on the Simple Question type rapidly. Therefore the performance of answering Simple Question fluctuates with the change of training sample size.

On the other hand, for the other categories, it could be found that the model can use data efficiently and obtain the best result by training on only 1% samples.

More samples might help the model on some question categories, but more training time is consumed. The training time of the REINFORCE module is plotted in Fig. 4. As can be seen, there is a significant increase in training time when training data increases from 1.0% to 1.2%. Together with the trend of the macro F1 value, as shown in Fig. 3, we can empirically determine the best trade-off between model performance and training efficiency at 1.0%.

5. Qualitative analysis

In this section, we analyze the quality of our NS-CQA model in more detail. We first present some success cases where NS-CQA can predict the action sequence that produces correct answers. A detailed analysis of typical errors is then performed, which sheds light on the areas that can be further investigated.

5.1. Sample cases

In Table 6, we present some example questions from different categories that our model NS-CQA can correctly predict action sequences.

We can inspect the complexity of the CQA problem from these instances. As is demonstrated in the table, Simple and Logical questions are simplest to answer since commonly, only 2–3 actions are needed. The following four categories, i.e., Quantitative, Comparative, Verification, and Quantitative Count, are relatively more difficult types with around 3–4 operations. For instance, the Verification question "Is Alda Pereira-Lemaitre a citizen of France and Emmelsbüll-Horsbüll?" has an answer "YES and NO respectively". Answering this question involves selecting all countries which Alda Pereira-Lemaitre is a citizen of, and verifying whether France and Emmelsbüll-Horsbüll is in this set respectively. The last type, Comparative Count, is the most complex for questions of which will be transformed into more than five actions.

Evident from the Quantitative Count and the Comparative Count questions in the last two rows of the table, answering CQA questions involve discrete actions. In the case of the question "How many assemblies or courts have control over the jurisdiction of the Free Hanseatic City of Bremen?", the set operation Union is required. In the case of the question "How many art genres express more number of human or concepts than floral painting?", numerical operations (GreaterThan and Count) are required.

These example questions attest to the challenging nature of the CQA dataset and the capability of our NS-CQA model.

5.2. Error analysis

To analyze the limitations of our NS-CQA model, 200 samples in each category that produce incorrect answers are randomly selected from the test dataset. In summary, a large number of errors can be categorized into one of the following five classes.

5.2.1. Linking problem

Since different entities/types might have the same surface name, in addition to literal similarity, the embedding of the description of entities/types and the embedding of question is employed to compute semantic similarity in our approach. When mapping the predicates to queries, a state-of-the-art convolutional sequence to sequence (Seq2Seq) learning model [40] implemented in fairSeq [43] is used. Even so, some linking problems remain.

Example:. "Where are around the same number of geographic locations located on as Big Salmon Range?".

When our model is answering the above question, the relation 'located on street' is wrongly linked to the question instead of the correct relation 'located on terrain feature'. This type of errors can be addressed by learning better semantic meaning of the entities/types/relations from the context in the knowledge graph.

5.2.2. Infeasible action

NS-CQA occasionally produces meaningless and repetitive actions which are semantically incorrect. For instance, some actions are predicted to union the same set, which is reluctant. In some cases, two repeated 'Count' actions are predicted.

Example:. "What social groups had Canada and Austria as their member?"

When our model is solving the above question, it predicts the following action sequence:

Select (Canada, member of, social group)

Bool (Austria)

EOQ

The operator 'Bool' is invalid since in this question the expected answer type is entities but not Boolean values. Semantic-based constraints could be employed to make the model produce feasible actions.

5.2.3. Spurious problem

In our approach, the pseudo-gold action sequences are generated by a BFS algorithm. Therefore corresponding to each question, multiple possible sequences may evaluate to the same expected results. Among these sequences, there might be some spurious action sequences. When training the model with such action sequences, the model may be misled and produce incorrect actions.

Example:. "Which musical ensembles were formed at Belfast?"

Our model transforms the above question into the following action sequence:

Select(Belfast, location, musical ensemble)

Inter(Belfast, location, musical ensemble)

EOQ

The second action 'Inter' is unnecessary to this question. Rule-based constraints could be incorporated to restrict the search process to meaningful actions.

5.2.4. Order of arguments

To decide the order of the entities/types in the actions is a difficult problem. For actions 'Select', 'Inter', 'Diff', and 'Union', the order of the arguments is decided by the following rule: the first argument in a triple pattern is related to the entity, and the last argument is associated with the type. In most cases, a sequence of entities/types in actions follows the order they appear in the question. Though our model is also trained to handle the situation that the sequence of entities/types does not appear in the same order, in some cases, the model is confused about which order to follow.

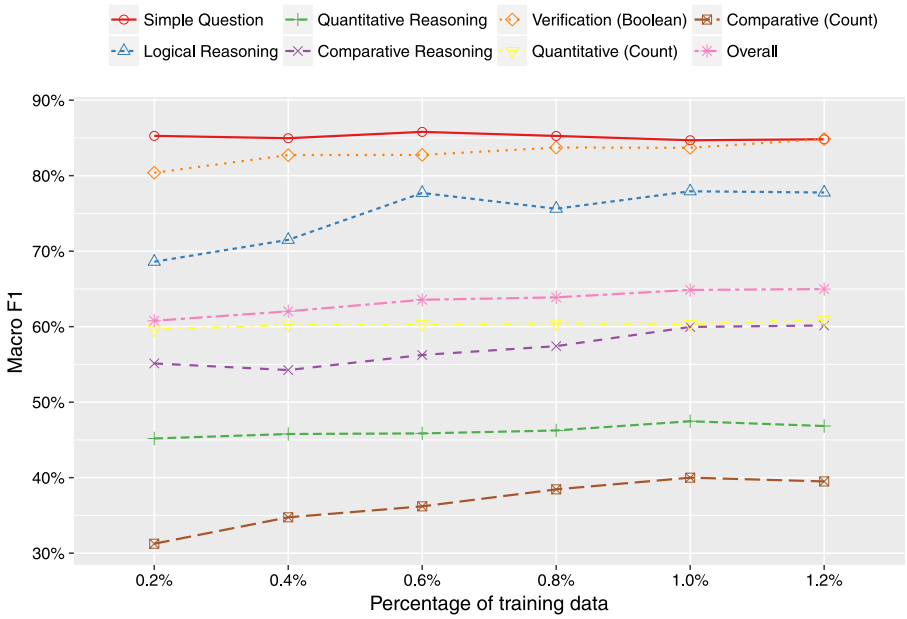

Fig. 3. Changes in macro F1 values with varying percentages of training data for the PG module.

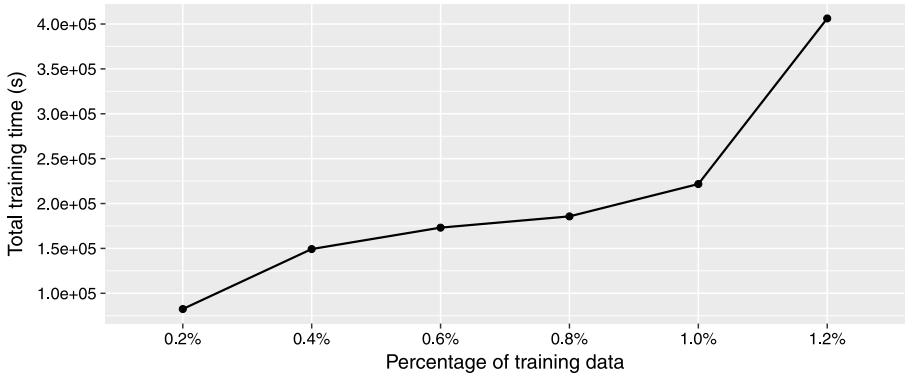

Fig. 4. Changes in training time (in seconds) with varying percentages of training data.

Example:. "Is Bernhard II, Duke of Saxe-Jena a child of William, Duke of Saxe-Weimar?"

Our model transforms the above question into the following action sequence:

Select (Bernhard II. Duke of Saxe-Jena, child, common name)

Bool (William. Duke of Saxe-Weimar)

EOQ

However, the correct action sequence should be the following. As can be seen, the order of the two entities is wrong in the predicted sequence.

Select (William. Duke of Saxe-Weimar, child, common name)

Bool (Bernhard II. Duke of Saxe-Jena)

EOQ

In future work, we will investigate whether incorporating more positional information can help alleviate this problem.

5.2.5. Approximation-related problem

The action 'Almost' is used to find the set of entities whose number is approximately the same as a given value, and such operation appears in the following four categories of questions: Quantitative Reasoning, Quantitative Count, Comparative Reasoning, and Comparative Count. The questions involving the 'Almost' action account for 4% of the total test dataset. When solving such questions, the range of the approximate interval is naturally vague. We define the following ad-hoc rule to address this vagueness: suppose we are required to find the value around N, when N is no larger than 5, the interval is $[N-1, N+1]$; when N is more significant than 5, the range is $[N-5, N+5]$. In some cases, this rule works, but in others not.

Example:. "Which political territories have diplomatic relations with approximately 14 administrative territories?"

The following action sequence could be produced to solve such questions:

SelectAll (political territorial entity, diplomatic relation, administrative territorial entity)

Almost (14)

EOQ

Following our rule, the approximate interval here should be [9, 19]. However, the correct answer (political territorial entities) may have several administrative territories outside this range.

Table 6
Examples of action sequences correctly predicted by NS-CQA for different types of questions.

Q. type	Question	KB artifacts	Action sequence	Answer
Simple	Which administrative territory is Danilo Ribeiro an inhabitant of?	**E1:** Danilo Ribeiro **R1:** country of citizenship **T1:** administrative territory	Select (E1, R1, T1) EOQ	Brazil
Logical	Which administrative territories are twin towns of London but not Bern?	**E1:** London **E2:** Bern **R1:** twinned adm. body **T1:** administrative territory	Select (E1, R1, T1) Diff (E2, R1, T1) EOQ	Sylhet, Tokyo, Podgorica, Phnom Penh, Delhi, Los Angeles, Sofia, New Delhi, …
Quantitative	Which sports teams have min number of stadia or architectural structures as their home venue?	**R1:** home venue **T1:** sports team **T2:** stadium **T3:** architectural structure	SelectAll (T1, R1, T2) SelectAll (T1, R1, T3) ArgMin () EOQ	Detroit Tigers, Drbak-Frogn IL, Club Sport Emelec, Chunichi Dragons, …
Comparative	Which buildings are a part of lesser number of architectural structures and universities than Midtown Tower?	**E1:** Midtown Tower **R1:** part of **T1:** building **T2:** architectural structure **T3:** university	SelectAll (T1, R1, T2) SelectAll (T1, R1, T3) LessThan (E1) EOQ	Amsterdam Centraal, Hospital de Sant Pau, Budapest Western Railway Terminal, El Castillo, …
Verification	Is Alda Pereira-Lemaitre a citizen of France and Emmelsb̀ull-Horsb́ull?	**E1:** Alda Pereira-Lemaitre **E2:** France **E3:** Emmelsb̀ull-Horsb́ull **R1:** country of citizenship **T1:** administrative territory	Select (E1, R1, T1) Bool (E2) Bool (E3) EOQ	YES and NO respectively
Quantitative Count	How many assemblies or courts have control over the jurisdiction of Free Hanseatic City of Bremen?	**E1:** Bremen **R1:** applies to jurisdiction **T1:** deliberative assembly **T2:** court	Select (E1, R1, T1) Union (E1, R1, T2) Count () EOQ	2
Comparative Count	How many art genres express more number of human or concepts than floral painting?	**E1:** floral painting **R1:** depicts **T1:** art genre **T2:** human **T3:** concept	SelectAll (T1, R1, T2) SelectAll (T1, R1, T3) GreaterThan (E1) Count () EOQ	8

Thus, the unfixed approximate interval may impair the performance of our model. We can manually tweak the rule of deciding the approximate interval. However, we emphasize that our model aims to show a robust framework to solve complex questions, but not to guess rules for approximation.

6. Conclusion

Answering complex questions on KBs is a challenging problem as it requires a model to perform discrete operations over KBs. State-of-the-art techniques combine neural networks and symbolic execution to address this problem. While practical, the challenges of these techniques reside in data-inefficiency, reward sparsity, and ample search space.

In this paper, we propose a data-efficient neural-symbolic model for complex KBQA that combines simple yet effective techniques, addressing some of the above deficiencies.

Firstly, we augment the model with a memory buffer. When the memory buffer maintains the generated successful trials for each training question, it will guide the model to replay and resample the promising trials more frequently, thus mitigating the data-inefficiency problem.

Secondly, by comparing the generated trials with the trials stored in the memory, we assign a bonus to the reward, which is the combination of proximity and novelty. Also, we propose an adaptive reward function. The reward bonus and the adaptive reward reshape the sparse reward into dense feedback that can efficiently guide policy optimization. Employing the curriculum-learning scheme, we gradually increase the proportion of proximity while decreasing the weight of novelty. By doing this, we encourage the model to find new trials while remembering the past successful trials.

Thirdly, we incorporate the copy and masking mechanisms in the model, and carefully design a set of primitive actions, to drastically reduce the size of the decoder output vocabulary by orders of magnitude. This significant reduction improves not only training efficiency but also model generalizability. Also, our actions free the model from the need to maintain complex intermediate memory modules, thus simplifies network design.

We conduct experiments on two challenging datasets on complex question answering. In comparison with three state-of-the-art techniques, our model achieves the best performance and significantly outperforming them in both the datasets.

CRediT authorship contribution statement

Yuncheng Hua: Conceptualization, Methodology, Software, Validation, Investigation, Writing - original draft. **Yuan-Fang Li:** Conceptualization, Writing - original draft, Writing - review & editing, Supervision. **Guilin Qi:** Writing - review & editing, Supervision, Project administration, Funding acquisition. **Wei Wu:** Software, Validation, Data curation. **Jingyao Zhang:** Software, Data curation. **Daiqing Qi:** Software, Data curation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was partially supported by the National Key Research and Development Program of China under grants (2018YFC0830200), the Natural Science Foundation of China grants (U1736204, 61602259), the Judicial Big Data Research Centre, China, School of Law at Southeast University, China, and the project no. 31511120201 and 31510040201.

References

[1] J. Berant, A. Chou, R. Frostig, P. Liang, Semantic parsing on freebase from question-answer pairs, in: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, 2013, pp. 1533–1544.

[2] X. Yao, B. Van Durme, Information extraction over structured data: Question answering with freebase, in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Vol. 1, 2014, pp. 956–966.

[3] W.-t. Yih, X. He, C. Meek, Semantic parsing for single-relation question answering, in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Vol. 2, 2014, pp. 643–648.

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

[5] W.-t. Yih, M.-W. Chang, X. He, J. Gao, Semantic parsing via staged query graph generation: Question answering with knowledge base, in: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Vol. 1, 2015, pp. 1321–1331.

[6] A. Saha, V. Pahuja, M.M. Khapra, K. Sankaranarayanan, S. Chandar, Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[7] A. Saha, G.A. Ansari, A. Laddha, K. Sankaranarayanan, S. Chakrabarti, Complex program induction for querying knowledge bases in the absence of gold programs, Trans. Assoc. Comput. Linguist. 7 (2019) 185–200.

[8] C. Liang, J. Berant, Q. Le, K.D. Forbus, N. Lao, Neural symbolic machines: Learning semantic parsers on freebase with weak supervision, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Vol. 1, 2017, pp. 23–33.

[9] G.A. Ansari, A. Saha, V. Kumar, M. Bhambhani, K. Sankaranarayanan, S. Chakrabarti, Neural program induction for KBQA without gold programs or query annotations, in: Proceedings of the 28th International Joint Conference on Artificial Intelligence, AAAI Press, 2019, pp. 4890–4896.

[10] N. Savinov, A. Raichuk, D. Vincent, R. Marinier, M. Pollefeys, T.P. Lillicrap, S. Gelly, Episodic curiosity through reachability, in: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, la, USA, May 6–9, 2019, OpenReview.net, 2019, URL https://openreview.net/forum?id=SkeK3s0qKQ.

[11] C. Liang, M. Norouzi, J. Berant, Q.V. Le, N. Lao, Memory augmented policy optimization for program synthesis and semantic parsing, in: Advances in Neural Information Processing Systems, 2018, pp. 9994–10006.

[12] D. Vrandečić, M. Krötzsch, Wikidata: a free collaborative knowledgebase, Commun. ACM 57 (10) (2014) 78–85.

[13] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, J. Taylor, Freebase: a collaboratively created graph database for structuring human knowledge, in: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, 2008, pp. 1247–1250.

[14] J. Gu, Z. Lu, H. Li, V.O. Li, Incorporating copying mechanism in sequence-to-sequence learning, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Vol. 1, 2016, pp. 1631–1640.

[15] Y. Bengio, J. Louradour, R. Collobert, J. Weston, Curriculum learning, in: Proceedings of the 26th Annual International Conference on Machine Learning, ICML'09, 2009, pp. 41–48.

[16] M. Fang, T. Zhou, Y. Du, L. Han, Z. Zhang, Curriculum-guided hindsight experience replay, in: Advances in Neural Information Processing Systems, 2019, pp. 12602–12613.

[17] W.-t. Yih, M. Richardson, C. Meek, M.-W. Chang, J. Suh, The value of semantic parse labeling for knowledge base question answering, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Vol. 2, 2016, pp. 201–206.

[18] J. Lehmann, T. Furche, G. Grasso, A.-C.N. Ngomo, C. Schallhart, A. Sellers, C. Unger, L. Buhmann, D. Gerber, K. Hoffner, et al., DEQA: deep web extraction for question answering, in: International Semantic Web Conference, Springer, 2012, pp. 131–147.

[19] J. Bao, N. Duan, M. Zhou, T. Zhao, Knowledge-based question answering as machine translation, in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Vol. 1, 2014, pp. 967–976.

[20] S. Hu, L. Zou, X. Zhang, A state-transition framework to answer complex questions over knowledge base, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 2018, pp. 2098–2108.

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

[22] X. He, D. Golub, Character-level question answering with attention, in: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 2016, pp. 1598–1607.

[23] D. Lukovnikov, A. Fischer, J. Lehmann, S. Auer, Neural network-based question answering over knowledge graphs on word and character level, in: Proceedings of the 26th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, 2017, pp. 1211–1220.

[24] A. Kumar, O. Irsoy, P. Ondruska, M. Iyyer, J. Bradbury, I. Gulrajani, V. Zhong, R. Paulus, R. Socher, Ask me anything: Dynamic memory networks for natural language processing, in: International Conference on Machine Learning, 2016, pp. 1378–1387.

[25] A. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, J. Weston, Key-value memory networks for directly reading documents, in: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 2016, pp. 1400–1409.

[26] K. Xu, Y. Lai, Y. Feng, Z. Wang, Enhancing key-value memory neural networks for knowledge based question answering, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 2019, pp. 2937–2947.

[27] K. Luo, F. Lin, X. Luo, K. Zhu, Knowledge base question answering via encoding of complex query graphs, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 2018, pp. 2185–2194.

[28] D. Guo, D. Tang, N. Duan, M. Zhou, J. Yin, Dialog-to-action: Conversational question answering over a large-scale knowledge base, in: Advances in Neural Information Processing Systems, 2018, pp. 2946–2955.

[29] S. Reddy, M. Lapata, M. Steedman, Large-scale semantic parsing without question-answer pairs, Trans. Assoc. Comput. Linguist. 2 (2014) 377–392.

[30] A. Bordes, J. Weston, N. Usunier, Open question answering with weakly supervised embedding models, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2014, pp. 165–180.

[31] A. Graves, G. Wayne, I. Danihelka, Neural turing machines, 2014, arXiv preprint arXiv:1410.5401.

[32] W. Zaremba, I. Sutskever, Reinforcement learning neural turing machines-revised, 2015, arXiv preprint arXiv:1505.00521.

[33] A. Neelakantan, Q.V. Le, I. Sutskever, Neural programmer: Inducing latent programs with gradient descent, 2015, arXiv preprint arXiv:1511.04834.

[34] P. Pasupat, P. Liang, Compositional semantic parsing on semi-structured tables, in: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Vol. 1, 2015, pp. 1470–1480.

[35] K. Guu, P. Pasupat, E. Liu, P. Liang, From language to programs: Bridging reinforcement learning and maximum marginal likelihood, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Vol. 1, 2017, pp. 1051–1062.

[36] K. Yi, J. Wu, C. Gan, A. Torralba, P. Kohli, J. Tenenbaum, Neural-symbolic vqa: Disentangling reasoning from vision and language understanding, in: Advances in Neural Information Processing Systems, 2018, pp. 1039–1050.

[37] A. Saha, G.A. Ansari, A. Laddha, K. Sankaranarayanan, S. Chakrabarti, Complex program induction for querying knowledge bases in the absence of gold programs, Trans. Assoc. Comput. Linguist. 7 (2019) 185–200.

[38] Z. Huang, W. Xu, K. Yu, Bidirectional LSTM-CRF models for sequence tagging, 2015, arXiv preprint arXiv:1508.01991.

[39] W. Yin, M. Yu, B. Xiang, B. Zhou, H. Schütze, Simple question answering by attentive convolutional neural network, in: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, 2016, pp. 1746–1756.

[40] J. Gehring, M. Auli, D. Grangier, D. Yarats, Y.N. Dauphin, Convolutional sequence to sequence learning, in: Proceedings of the 34th International Conference on Machine Learning, Vol. 70, JMLR. org, 2017, pp. 1243–1252.

[41] R.J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, Mach. Learn. 8 (3–4) (1992) 229–256.

[42] S.J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, V. Goel, Self-critical sequence training for image captioning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 7008–7024.

[43] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, M. Auli, fairseq: A fast, extensible toolkit for sequence modeling, in: Proceedings of NAACL-HLT 2019: Demonstrations, 2019.

Chapter 4

Complex Knowledge Base Question Answering via Meta Reinforcement Learning

In Chapter 3, we have introduced a novel NPI-based method that alleviates several challenges that inherently lie in existing NPI-based CQA systems. Like previous work, we trained a monolithic model, i.e., a model that uses one set of parameters to fit all training samples to answer questions in the given dataset. Though proved to have better performance than other state-of-the-art models, our model still faces a problem. The process of finding a global, optimal set of parameter values that perform well on a wide range of different questions is arduous. When conducting experiments, we found that the monolithic model always behaved well on some types of questions while achieved a much worse performance on other questions, leading to uneven performance. Therefore, using a monolithic, one-size-fits-all model may not be the best strategy for the CQA task, as the questions and their corresponding logical forms in the given dataset could vary widely. For instance, comparing the two examples "What is China's capital?" and "How many countries have more rivers than China?", the different length and pattern of the questions could lead to corresponding programs with a substantial variety of structures ("Select(China, capital, city)" and "Select(China, flow, river), Count(), SelectAll(country, flow, river), GreaterThan(China), Count()", respectively).

Since a one-size-fits-all model is not perfectly applicable to certain cases where the style and format of the questions vary vastly, an alternative training paradigm is introduced: instead of finding

an optimal set of model parameters, learning multiple sets of parameters could be more effective, where each set is used exclusively for a particular *task*—a group of questions that share a quite similar pattern. To accomplish this, we change our objective: from finding the *global optimal* parameters to learning the *optimal initial* parameters, by which multiple sets of parameters could be generated adaptively for varied *tasks*.

Meta-learning, aka learning-to-learn, aims to train a model to learn the generic knowledge across a variety of tasks [161], such that the model could quickly adapt to a new task using only a few training samples of that task. A popular approach is to train a meta-learner to learn a high-level strategy generalizes across many different tasks and use the meta-learner to update the model's parameters for a novel task [160]. Previous work on the meta-learning frames the meta-learner in a single neural network [188, 189], increasing the number of parameters to train.

In contrast, instead of retraining a new neural network to control the update of the model's weights—which needs to optimize additional parameters—MAML [179], a meta-learning based approach is proposed to simplify the design of meta-learner. MAML views meta-learner as model's *initial parameters*, representing the generic knowledge applicable to multiple changing tasks. In their work, well-learned initial parameters enable the model to generalize a novel task by fine-tuning the initial parameters from a small number of similar tasks on the fly. MAML divides the meta-learning process into two stages: (i) meta-training, which retrieves (normally using a retriever) a support set (tasks similar to a given new task) to learn the task-specific parameters based on the initial parameters, and (ii) meta-testing, that updates the initial parameters based on evaluating the task-specific parameters.

Considering simplicity, several methods have extended the MAML framework for solving the one-size-fits-all problem in several semantic parsing tasks. Huang et al. [17] build a framework on MAML to solve a semantic parsing task, where a NLQ needs to be translated into a SQL query to access relational databases. To construct support sets for meta-training, Huang et al. design a relevance function that takes SQL query's format and question length into consideration. Similarly, Guo et al. [18] propose a MAML-based framework to solve a conversational question answering task, i.e., answering context-dependent questions based on a KB, where each question is part of a dialog. Guo et al. design a context-aware retriever that incorporates the conversational history to retrieve samples close to a given task in the latent space. Particularly, the retriever is trained using a supervised learning paradigm, where annotations are necessary for evaluation. However, both tasks' settings are different

from ours in that we consider answering single-turn (instead of multiple-round) questions over a KB (rather than a relational database). Also, in our problem setup, the annotations are unavailable in the given dataset, making it impossible to train the retriever under the supervised learning paradigm. Accordingly, we need to design a retriever in accordance with our CQA problem and thus build a meta-learning-based question answering framework to solve the CQA task.

In our work, we build a CQA framework upon MAML to accomplish the following objectives:

1. Design an unsupervised retriever to find similar questions as a support set.
2. Employ meta-learning to optimize the meta-learner, aka CQA model's initial parameters.
3. Use execution results of the logical forms as rewards to update initial parameters and finetune the task-specific parameters under a weak-supervised learning paradigm.

Therefore, we propose a Meta-Reinforcement Learning (Meta-RL) framework that acquires knowledge from similar tasks for fast adaptation to a novel task and updates the model's weights with policy gradients. We name it as **M**eta-**RL** approach for **C**omplex **Q**uestion **A**nswering (**MRL-CQA**). This research work has been published in the *2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*. The complete version of the manuscript is attached in the subsequent pages.

Hua, Y., Li, Y., Haffari, G., Qi, G. and Wu, T. Few-shot Complex Knowledge Base Question Answering via Meta Reinforcement Learning. In *2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*, November 16-20, 2020, Proceedings, pages 5827-5837.

Few-Shot Complex Knowledge Base Question Answering via Meta Reinforcement Learning

Yuncheng Hua[†,§], **Yuan-Fang Li**[◇], **Gholamreza Haffari**[◇], **Guilin Qi**[†,‡*] **and Tongtong Wu**[†]
[†]School of Computer Science and Engineering, Southeast University, China
[◇]Faculty of Information Technology, Monash University, Australia
[§]Southeast University-Monash University Joint Research Institute, China
[‡]Key Laboratory of Computer Network and Information Integration, Southeast University
[†]{devinhua, gqi, wutong8023}@seu.edu.cn
[◇]{yuanfang.li, gholamreza.haffari}@monash.edu

Abstract

Complex question-answering (CQA) involves answering complex natural-language questions on a knowledge base (KB). However, the conventional neural program induction (NPI) approach exhibits uneven performance when the questions have different types, harboring inherently different characteristics, e.g., difficulty level. This paper proposes a meta-reinforcement learning approach to program induction in CQA to tackle the potential distributional bias in questions. Our method quickly and effectively adapts the meta-learned programmer to new questions based on the most similar questions retrieved from the training data. The meta-learned policy is then used to learn a good programming policy, utilizing the *trial* trajectories and their rewards for similar questions in the support set. Our method achieves state-of-the-art performance on the CQA dataset (Saha et al., 2018) while using only five trial trajectories for the top-5 retrieved questions in each support set, and meta-training on tasks constructed from only 1% of the training set. We have released our code at https://github.com/DevinJake/MRL-CQA.

1 Introduction

Knowledge-base question-answering (KBQA) interrogates a knowledge-base (KB) (Yin et al., 2016; Yu et al., 2017; Jin et al., 2019) by interpreting natural-language questions as logical forms (*annotations*), which can be directly executed on the KB to yield answers (*denotations*) (Pasupat and Liang, 2016). KBQA includes *simple questions* that retrieve answers from single-hop triples ("what is Donald Trump's nationality") (Berant et al., 2013; Yih et al., 2014), *multi-hop questions* that infer answers over triple chains of at least 2 hops under specific constraints ("who is the president of the European Union 2012") (Yih et al., 2016; Liang et al., 2017), and *complex questions* that involve set operations ("how many rivers flow through India and China") (Saha et al., 2019). In particular, complex question answering (CQA) (Saha et al., 2018) is a sophisticated KBQA task in which a sequence of discrete *actions*—e.g., set intersection and union, counting, comparison—needs to be executed, and is the subject of this paper.

Consider the complex question "How many rivers flow through India and China?". We first form a set of entities whose type is river and flow in China from the KB. We then form another set for rivers that flow through India. The answer is then generated by *counting* the entities in the *intersection* of the two sets. More concretely, the question is transformed into the action sequence "Select (China, flow, river), Intersection (India, flow, river), Count", which is executed on the KB to yield the answer. As such, the CQA task results in a massive search space beyond just entities in the KB and includes (lists of) Boolean values and integers. Multi-hop questions only require the join operator. In contrast, CQA requires various types of additional symbolic reasoning, e.g., logical, comparative, and quantitative reasoning (Shen et al., 2019; Ansari et al., 2019), where a more diverse array of complex queries is involved (Saha et al., 2019). The massive search space and complex queries make CQA considerably challenging and more complicated than multi-hop question answering.

Due to the difficulty of collecting annotations, the existing CQA dataset (Saha et al., 2018) only contains the denotations for each question. The literature takes two approaches to deal with the missing annotations. The first approach aims to transform learning a CQA model into learning by demonstration, aka imitation learning, where a *pseudo-gold* action sequence is produced for the questions in the training set (Guo et al., 2018). This is done by employing a blind search algorithm, i.e.,

*Corresponding Author.

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 5827–5837,
November 16–20, 2020.

breadth-first search (BFS), to find a sequence of actions whose execution would yield the correct answer. This pseudo-gold annotation is then used to train the programmer using teacher forcing, aka behaviour cloning. However, BFS inevitably produces a single annotation and is ignorant to many other plausible annotations yielding the correct answer. To alleviate this issue, a second approach was proposed based on reinforcement learning (RL) to use the search policy prescribed by the programmer (Hua et al., 2020; Neelakantan et al., 2016; Liang et al., 2017). Compared to BFS which is a blind search algorithm, the RL-trained programmer can be regarded as an informed search algorithm for target programs. Therefore, the RL policy not only addresses the limitation of the 1-to-1 mapping between the questions and annotations, but also produces reasonable programs faster than BFS.

The conventional approach to CQA is to train *one* model to fit the entire training set, and then use it for answering all complex questions at the test time. However, such a one-size-fits-all strategy is sub-optimal as the test questions may have diversity due to their inherently different characteristics (Huang et al., 2018). For instance, in the CQA dataset, the samples could be categorized into seven different types, e.g., those capturing logical/comparative/quantitative reasoning. The length and complexity of questions in one group are likely to differ from those in other groups. Therefore, action sequences relevant to different groups may have significant deviations, and it is hard to learn a one-size-fits-all model that could adapt to varied types of questions. An exception is (Guo et al., 2019), which proposes a few-shot learning approach, i.e., S2A, to solve the CQA problem with a retriever and a meta-learner. The retriever selects similar instances from the training dataset to form *tasks*, and the meta-learner is trained on these tasks to learn how to quickly adapt to a new task created by the target question of interest at the test time. However, Guo et al. (2019) make use of teacher forcing within the learning by demonstration approach, which suffers from the aforementioned drawbacks. Also, though S2A is the most similar to ours, the tasks are very different. S2A aims to answer *context-dependent* questions, where each question is part of a *multiple-turn* conversation. On the contrary, we consider the different task where the questions are *single-turn* and have no context. Thus, a novel challenge arises in retrieving accurate support sets without conversation-based context information.

In this paper, we propose a Meta-RL approach for CQA (MRL-CQA), where the model adapts to the target question by *trials* and the corresponding reward signals on the retrieved instances. In the meta-learning stage, our approach learns a RL policy across the tasks for both (i) collecting trial trajectories for effective learning, and (ii) learning to adapt programmer by effectively combining the collected trajectories.

The accumulated general knowledge acquired during meta-learning enables the model to generalize over varied tasks instead of fitting the distribution of data points from a single task. Thus, the tasks generated from tiny (less than 1%) portion of the training data are sufficient for meta learner to acquire the general knowledge. Our method achieves state-of-the-art performance on the CQA dataset with overall macro and micro F1 scores of 66.25% and 77.71%, respectively.

2 Meta-RL for Complex Question Answering

The problem we study in this paper is transforming a complex natural-language question into a sequence of actions, i.e., a sequence-to-sequence learning task. By executing the actions, relevant triples are fetched from the KB, from which the answer to the question is induced. We tackle this problem with few-shot meta reinforcement learning to decrease the reliance on data annotation and increase the accuracy for different questions.

Let $\boldsymbol{q}$ denote the input sequence, including the complex question and the KB artifacts, i.e., entities, relations, and types in KB that are relevant to the problem. Let $\boldsymbol{\tau}$ denotes the output sequence, i.e., an action sequence that the agent generates to answer the question. Let $R(\boldsymbol{\tau}|\boldsymbol{q}) \in [0, 1]$ denotes the partial reward feedback that tells whether or not the action sequence yields the correct answer. To simplify the notation, we denote the reward function by $R(\boldsymbol{\tau})$. The training objective is to maximize the expected reward by optimizing the parameter $\boldsymbol{\theta}$ of the policy $\pi(\boldsymbol{\tau}|\boldsymbol{q}; \boldsymbol{\theta})$, i.e., improving the accuracy of the policy in answering unseen questions. For the test, the agent needs to generate an action sequence $\boldsymbol{\tau}^*$ for the input sequence using a search algorithm, e.g., greedy decoding, which is then executed on KB to get the answer.

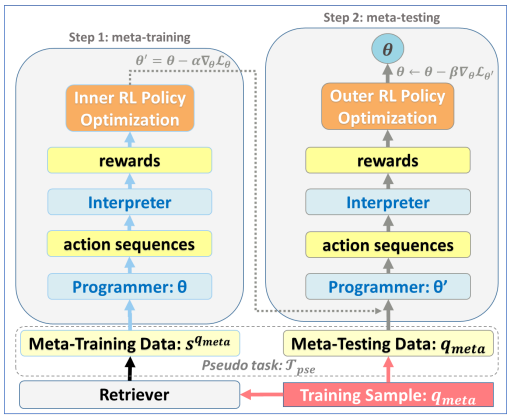

Figure 1: The high-level architecture of our approach.

2.1 Overview of the Framework

Our framework for few-shot learning of CQA is illustrated in Figure 1. In our framework, we view each new training question as the test sample of a pseudo task, and we aim to learn a specific model devoted to solving the task. When faced with a question $\boldsymbol{q_{meta}}$, we first employ the retriever to find top-N samples $\boldsymbol{s^{q_{meta}}}$ in the training dataset, which are the most similar to $\boldsymbol{q_{meta}}$. We consider $\boldsymbol{s^{q_{meta}}}$ as meta-training data used to learn a particular model, and view the question $\boldsymbol{q_{meta}}$ as the meta-testing data to evaluate the model. Therefore, meta-training data $\boldsymbol{s^{q_{meta}}}$ and meta-testing data $\boldsymbol{q_{meta}}$ form a pseudo task $\boldsymbol{\mathcal{T}_{pse}}$.

In the meta-training step (Step 1 in Figure 1), the action sequences that correspond to $\boldsymbol{s^{q_{meta}}}$ will be generated based on the current parameter $\boldsymbol{\theta}$ of the *programmer*. The *interpreter* executes the action sequences and evaluates the generated answers to produce rewards. The rewards lead to gradient updates that finetune the current model to get a task-specific *programmer* with the parameter of $\boldsymbol{\theta'}$. After that, in the meta-testing step (Step 2 in Figure 1), the actions of $\boldsymbol{q_{meta}}$ are produced based on $\boldsymbol{\theta'}$ and are evaluated to update $\boldsymbol{\theta}$. The training approach is depicted in Algorithm 1. In both the meta-training and meta-testing steps, REINFORCE (Williams, 1992) is used to optimize the *programmer*.

Similarly, in the inference phase, we consider each test question as a new individual task. We retrieve top-N data points from the training dataset to form the meta-training data. Instead of applying the general *programmer* with $\boldsymbol{\theta}$ directly, the meta-training data is used to finetune a specific parameter $\boldsymbol{\theta'}$ that fits the test question and infer the output.

2.2 Programmer and Interpreter

Programmer Our *programmer* is a sequence-to-sequence (Seq2Seq) model. Given the input sequence $\boldsymbol{q}$ with tokens $(w_1, \ldots, w_M)$, the *programmer* produces actions $(a_1, \ldots, a_T)$. The input sequence is the original complex question concatenated with the KB artifacts appear in the query, and the output is the words or tokens. The output at each time step is a single token.

In the *programmer*, the encoder is a Long Short Term Memory (LSTM) network that takes a question of variable length as input and generates an encoder output vector $\boldsymbol{e}_i$ at each time step i as: $(\boldsymbol{e}_i, \boldsymbol{h}_i) = LSTM[\phi_E(w_i), \boldsymbol{h}_{i-1}]$. Here $\phi_E(w_i)$ is word embedding of token w_i, and $(\boldsymbol{e}_i, \boldsymbol{h}_i)$ is the output and hidden vector of the i-th time step. The dimension of $\boldsymbol{e}_i$ and $\boldsymbol{h}_i$ are set as the same.

Our decoder of the *programmer* is another attention-based LSTM model that selects output token a_t from the output vocabulary $\mathcal{V}_{output}$. The decoder generates a hidden vector $\boldsymbol{g}_t$ from the previous output token a_{t-1}. The previous step's hidden vector $\boldsymbol{g}_{t-1}$ is fed to an attention layer to obtain a context vector $\boldsymbol{c}_t$ as a weighted sum of the encoded states using the standard attention mechanism. The current step's $\boldsymbol{g}_t$ is generated via $\boldsymbol{g}_t = LSTM\{\boldsymbol{g}_{t-1}, [\phi_D(a_{t-1}), \boldsymbol{c}_t]\}$, where ϕ_D is the word embedding of input token a_{t-1}. The decoder state $\boldsymbol{g}_t$ is used to compute the score of the target word $v \in \mathcal{V}_{output}$ as,

$$\pi(a_t = v|\boldsymbol{a}_{<t}, \boldsymbol{q}) = \text{softmax}(\mathbf{W} \cdot \boldsymbol{g}_t + \boldsymbol{b})_v \quad (1)$$

where $\mathbf{W}$ and $\boldsymbol{b}$ are trainable parameters, and $\boldsymbol{a}_{<t}$ denotes all tokens generated before the time step t. We view all the weights in the *programmer* as the parameter $\boldsymbol{\theta}$, thus we have the probability that the *programmer* produces an action sequence $\boldsymbol{\tau}$ with tokens $\{a_1, ..., a_T\}$ as,

$$\pi(\boldsymbol{\tau}|\boldsymbol{q}; \theta) = \prod_{t=1}^{T} \pi(a_t = v|\boldsymbol{a}_{<t}, \boldsymbol{q}). \quad (2)$$

When adapting the policy to a target question, our *programmer* outputs action sequences following the distribution computed by equation 2. By treating decoding as a stochastic process, the *programmer* performs random sampling from the probability distribution of action sequences to increase the output sequences' diversity.

Interpreter After the *programmer* generates the entire sequence of actions, the *interpreter* executes

the sequence to produce an answer. It compares the predicted answer with the ground-truth answer and outputs a partial reward. If the type of the output answer is different from that of the ground-truth answer, the action sequence that generates this answer will be given a reward of 0. Otherwise, to alleviate the sparse reward problem, the *interpreter* takes the Jaccard score of the output answer set and the ground-truth answer set as the partial reward, and sends it back to update parameters of the *programmer* as the supervision signal.

2.3 Meta Training and Testing

We formulate training of the programmer in a RL setting, where an agent interacts with an environment in discrete time steps. At each time step t, the agent produces an action (in our case a word/token) a_t sampled from the policy $\pi(a_t|\boldsymbol{a}_{<t}, \boldsymbol{q}; \boldsymbol{\theta})$, where $\boldsymbol{a}_{<t}$ denotes the sequence generated by the agent from step 1 to $t-1$, and $\boldsymbol{q}$ is the input sequence. The policy of the agent is the *programmer*, i.e., LSTM-attention model M with parameter $\boldsymbol{\theta}$. The natural-language question concatenated with the KB artifacts will be fed into the encoder as an input, and a sequence of actions is output from the decoder. In our work, we regard each action sequence produced by the model as one trajectory. The action sequence is therefore executed to yield a generated answer, and the similarity between the output answer with the ground-truth answer is then computed. The environment considers the similarity as the reward R corresponding to the trajectory $\boldsymbol{\tau}$ and gives it back to the agent. In standard RL, the parameter of the policy $\boldsymbol{\theta}$ is updated to maximize the expected reward, $\mathbb{E}_{\boldsymbol{\tau}\sim\pi(\boldsymbol{\tau}|\boldsymbol{q};\boldsymbol{\theta})}[R(\boldsymbol{\tau})]$.

In our work, answering each question in the training dataset is considered as an individual task, and a model adaptive to a new task is learned from the support set questions. To make the meta-learned model generalize to all unseen tasks, we sample the tasks following the distribution of tasks in the training dataset. We first sample a small subset of the questions Q_{meta} from the training dataset and expand the questions into tasks $\mathcal{T}_{meta}$ through retrieving the top-N samples, then extract a batch of tasks $\mathcal{T}'$ from $\mathcal{T}_{meta}$ under the distribution of tasks in $\mathcal{T}_{meta}$ to update parameters.

To fully use the training dataset and decrease training time, we study how to train a competitive model by using as few training samples as possible. As we view CQA as a RL problem under few-shot learning conditions, we make use of Meta-RL techniques (Finn et al., 2017) to adapt the programmer to a new task with a few training samples. Meta-RL aims to meta-learn an agent that can rapidly learn the optimal policy for a new task $\mathcal{T}$. It amounts to learn optimized $\boldsymbol{\theta}^*$ using K trial trajectories and the rewards for the support set of a new task.

We use the gradient-based meta-learning method to solve the Meta-RL problem such that we can obtain the optimal policy for a given task after performing a few steps of vanilla policy gradient (VPG) (Williams, 1992; Sutton et al., 2000). We divide the meta-learning process into two steps to solve a task, namely the meta-training step and the meta-testing step. Suppose we are trying to solve the pseudo-task $\mathcal{T}_{pse}$, which consists of N meta-training questions $\boldsymbol{s}^{\boldsymbol{q}_{meta}}$ that are the most similar to the meta-testing sample q_{meta}. The model first generates K trajectories for each question in $\boldsymbol{s}^{\boldsymbol{q}_{meta}}$ based on $\boldsymbol{\theta}$. The reward of each trajectory is given by the environment and then subsequently used to compute $\boldsymbol{\theta}'$ adapted to task $\mathcal{T}_{pse}$, as

$$\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta} + \eta_1 \nabla_{\boldsymbol{\theta}} \sum_{\boldsymbol{q}\in\boldsymbol{s}^{\boldsymbol{q}_{meta}}} \mathbb{E}_{\boldsymbol{\tau}\sim\pi(\boldsymbol{\tau}|\boldsymbol{q};\theta)}[R(\boldsymbol{\tau})] \quad (3)$$

During meta-testing, another K' trajectories corresponding to question $\boldsymbol{q}_{meta}$ are further produced by $\boldsymbol{\theta}'$. The reward of K' trajectories are considered as the evaluation of the adapted policy $\boldsymbol{\theta}'$ for the given task $\mathcal{T}_{pse}$; thus we have the objective,

$$J(\boldsymbol{\theta}') \stackrel{\text{def}}{=} \mathbb{E}_{\boldsymbol{\tau}'\sim\pi(\boldsymbol{\tau}'|\boldsymbol{q}_{meta};\boldsymbol{\theta}')}[R(\boldsymbol{\tau}')] \quad (4)$$

The parameter of the generic policy $\boldsymbol{\theta}$ are then trained by maximising the objective $J(\boldsymbol{\theta}')$,

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta_2 \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}') \quad (5)$$

In each VPG step, since we have N samples in $\boldsymbol{s}^{\boldsymbol{q}_{meta}}$, we use N policy gradient adaptation steps to update $\boldsymbol{\theta}'$. Meanwhile, we use one policy gradient step to optimize θ based on the evaluation of $\boldsymbol{\theta}'$. Monte Carlo integration is used as the approximation strategy in VPG (Guu et al., 2017). We summarise the meta-learning approach in Alg.1.

When making inferences, for each question $\boldsymbol{q}_{test}$, the retriever creates a pseudo-task, similar to the meta-learning process. The top-N similar questions to $\boldsymbol{q}_{test}$ form the support set $\boldsymbol{s}^{\boldsymbol{q}_{test}}$, and are used to obtain the adapted model $\boldsymbol{\theta}^{*'}$, starting from the meta learned policy $\boldsymbol{\theta}^*$. The adapted model is then used to generate the program and compute the target question's final answer.

Algorithm 1: Meta-RL (training time)

Input: Dataset Q_{train}, step size η_1, η_2
Output: Meta-learned policy $\boldsymbol{\theta}^*$

Randomly initialize $\boldsymbol{\theta}$
Randomly sample $Q_{meta} \sim Q_{train}$
Expand $Q_{meta} \rightarrow \mathcal{T}_{meta}$
while *not done* **do**
  Sample a batch of tasks $\mathcal{T}' \sim \mathcal{T}_{meta}$
  for $\mathcal{T}_{pse} \in \mathcal{T}'$ **do**
    $\mathcal{L} \leftarrow 0$
    for *each question* $\boldsymbol{q} \in \boldsymbol{s}^{\boldsymbol{q}_{meta}}$ **do**
      Sample K trajectories: $\boldsymbol{\tau}_k \sim \pi(\boldsymbol{\tau}|\boldsymbol{q};\boldsymbol{\theta})$
      $\mathcal{L} \leftarrow \mathcal{L} + \frac{1}{K}\sum_{k=1}^{K} R(\boldsymbol{\tau_k}) logp_{\boldsymbol{\theta}}(\boldsymbol{\tau_k})$
    $\boldsymbol{\theta'} \leftarrow \boldsymbol{\theta} + \eta_1 \nabla_{\boldsymbol{\theta}} \mathcal{L}$
    Sample K' trajectories: $\boldsymbol{\tau}_{k'} \sim \pi(\boldsymbol{\tau}|\boldsymbol{q}_{meta};\boldsymbol{\theta'})$
    $J_{\boldsymbol{q}_{meta}}(\boldsymbol{\theta'}) \leftarrow \frac{1}{K'}\sum_{k'=1}^{K'} R(\boldsymbol{\tau}_{k'}) logp_{\boldsymbol{\theta'}}(\boldsymbol{\tau}_{k'})$
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta_2 \nabla_{\boldsymbol{\theta}} \sum_{\mathcal{T}_{pse} \in \mathcal{T}'} J_{\boldsymbol{q}_{meta}}(\boldsymbol{\theta'})$
Return The meta-learned policy $\boldsymbol{\theta}^* \leftarrow \boldsymbol{\theta}$

2.4 Question Retriever

We propose an unsupervised retriever that finds, from the training dataset, relevant support samples for the tasks in both the training and test phases. We propose a relevance function that measures the similarity between two questions in two aspects: (1) the number of KB artifacts (i.e., entities, relations, and types) in the questions and (2) question semantic similarity.

If the two questions have the same number of KB artifacts, the structure of their corresponding action sequences are more likely to be resembled. We calculate the similarity in terms of the number of entities of two questions $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ by $sim_e(\boldsymbol{q}_1, \boldsymbol{q}_2) = 1 - \frac{|q_e(\boldsymbol{q}_1) - q_e(\boldsymbol{q}_2)|}{\max(q_e(\boldsymbol{q}_1), q_e(\boldsymbol{q}_2))}$. The function $q_e(\boldsymbol{q})$ counts the number of entities in the question. Similarly, we compute the similarities in terms of relations and types in the same way with $sim_r(\boldsymbol{q}_1, \boldsymbol{q}_2)$ and $sim_t(\boldsymbol{q}_1, \boldsymbol{q}_2)$ respectively. The KB artifact similarity $sim_a(\boldsymbol{q}_1, \boldsymbol{q}_2)$ is computed by the product of the above three similarities.

For two questions, the more common words they have, the more semantically similar they are. Based on this intuition, we propose a semantic similarity function based on Jaccard similarity in an unsupervised way. Suppose there is a set of i words $\{w_1^1, ..., w_1^i\}$ in $\boldsymbol{q}_1$ and j words $\{w_2^1, ..., w_2^j\}$ in $\boldsymbol{q}_2$, and word similarity $sim(w^i, w^j)$ is calculating using the Cosine similarity.

For each word in $\boldsymbol{q}_1$, we first collect the word pairs from the words in $\boldsymbol{q}_2$, whose highest similarity exceeds a pre-defined threshold value. We denote with $sem_{int}(\boldsymbol{q}_1, \boldsymbol{q}_2)$ the sum of similarity values of the word pairs:

$$sem_{int}(\boldsymbol{q}_1, \boldsymbol{q}_2) = \sum_{m=1}^{i} (\max_{n=1}^{j}(sim(w_1^m, w_2^n))) \tag{6}$$

After removing this set of highly similar words from the two questions, we denote the remaining tokens as $\{w_1^{remain}\}$ and $\{w_2^{remain}\}$, which represent the different parts of the two questions. We sum up the embeddings of the words in $\{w_1^{remain}\}$ as $\mathbf{w}_1^{remain}$, and compute $\mathbf{w}_2^{remain}$ in the same way. The function $sem_{diff}(\boldsymbol{q}_1, \boldsymbol{q}_2)$ measures how different $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ are:

$$sem_{diff}(\boldsymbol{q}_1, \boldsymbol{q}_2) = \max(|\{w_1^{remain}\}|, |\{w_2^{remain}\}|) * (1 - sim(\mathbf{w}_1^{remain}, \mathbf{w}_2^{remain})), \tag{7}$$

where $|\{w\}|$ returns the cardinality of the set $\{w\}$.

We define the semantic similarity between $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ as: $sim_s(\boldsymbol{q}_1, \boldsymbol{q}_2) = \frac{sem_{int}(\boldsymbol{q}_1, \boldsymbol{q}_2)}{sem_{int}(\boldsymbol{q}_1, \boldsymbol{q}_2) + sem_{diff}(\boldsymbol{q}_1, \boldsymbol{q}_2)}$, and therefore calculate the similarity between $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ with $sim_a(\boldsymbol{q}_1, \boldsymbol{q}_2) * sim_s(\boldsymbol{q}_1, \boldsymbol{q}_2)$.

3 Experiments

In this section, we present the empirical evaluation of our MRL-CQA framework.

Dataset. We evaluated our model on the large-scale CQA (Complex Question Answering) dataset (Saha et al., 2018). Generated from the Wikidata KB (Vrandecic and Krötzsch, 2014), CQA contains 944K/100K/156K QA pairs for training, validation, and testing, respectively. In the CQA dataset, each QA pair consists of a complex, natural-language question and the corresponding ground-truth answer (i.e., denotation). We note that annotations, i.e., gold action sequences related to the questions, are not given in the CQA dataset. The CQA questions are organized into seven categories of different characteristics, as shown in the Table 1. Some categories have entities as answers (e.g., "Simple Question"), while others have (lists of) numbers (e.g., "Quantitative (Count)")

or Booleans (e.g., "Verification (Boolean)") as answers. The size of different categories in CQA is uneven. The number of instances in each category in the training set is 462K, 93K, 99K, 43K, 41K, 122K, and 42K for Simple Question, Logical Reasoning, Quantitative Reasoning, Verification (Boolean), Comparative Reasoning, Quantitative (Count), and Comparative (Count), respectively.

Based on the length of the induced programs and performance of the best models, we further organized the seven categories into two groups: *easy*—the first four categories, and *hard*—the last three types, in Table 1. We used the same evaluation metrics employed in the original paper (Saha et al., 2018), the F1 measure, to evaluate models.

Training Configuration. In the CQA dataset, since the annotated action sequence are not provided, we randomly sampled 1% of the training set (approx. 10K out of 944K training samples) and followed (Guo et al., 2019) to annotate them with pseudo-gold action sequences by using a BFS algorithm. We denoted the 1% questions and relevant pseudo-gold action sequences as Q_{pre}. The Q_{pre} was used to train the LSTM-based programmer, which was further optimized through the Policy Gradient (PG) algorithm (Williams, 1992; Sutton et al., 2000) with another 1% unannotated questions from the training set. We denoted this model by **PG**, which is also a model variant proposed in (Hua et al., 2020). We trained the meta learner on another 2K training samples (Q_{meta} in Alg.1), representing only approx. 0.2% of the training set. This meta learner is our full model: **MRL-CQA**.

In our work, we chose the attention-based LSTM model instead of the Transformer (Vaswani et al., 2017) to design the *programmer*. We set the sizes of embedding and hidden units in our LSTM model as 50 and 128 respectively, thus the maximum number of the parameters in our model is about 1.2M. However, the base model of the Transformer (12 layers, 12 heads, and 768 hidden units) has 110M parameters (Wolf et al., 2019), which are much more than those of our model. Given the small size of the training samples and the weak supervision signal (reward in our work), it is harder to train the model with more parameters. Therefore we chose LSTM rather than the Transformer.

We implemented our model in PyTorch and employed the Reptile meta-learning algorithm to optimize the meta-learned policy (Nichol and Schulman, 2018). The weights of the model and the word embeddings were randomly initialized and further updated within the process of training. In meta-learning, we set $\eta_1 = 1e-4$ (Equation 3) and $\eta_2 = 0.1$ (Equation 5). We set $N = 5$ and threshold value at 0.85 when forming the support set. For each question, we generate five action sequences to output the answers. The Adam optimizer is used in RL to maximizes the expected reward.

Among the baseline models, we ran the open-source code of KVmem (Saha et al., 2018) and CIPITR (Saha et al., 2019) to train the model. As the code of NSM (Liang et al., 2017) has not been made available, we re-implemented it and incorporated our programmer to predict programs, and employed the reinforcement learning settings in NSM to optimize the programmer. When inferring the testing samples, we used the top beam, i.e., the predicted program with the highest probability in the beam to yield the answers. We presented the best result we got to compare the baseline models.

3.1 Model Comparisons

We evaluated our model, MRL-CQA, against three baseline methods on the CQA dataset: KVmem, NSM, and CIPITR. It must be pointed out that CIPITR separately trained *one single model for each of the seven question categories*. We denote the model learned in this way as **CIP-Sep**. CIPITR also trained *one single model over all categories of training instances* and used this single model to answer all questions. We denote this single model as **CIP-All**. We separately present the performance of these two variations of CIPITR in Table 1. On the contrary, we tuned MRL-CQA on all categories of questions with one set of model parameters.

Table 1 summarizes the performance in F1 of the six models on the test set of CQA, organised into seven question categories. We note that the first four categories (first four rows in Table 1) are relatively simple, and the last three (middle three rows) are more challenging. We also report the overall macro and micro F1 values (last two rows).

As can be seen, our MRL-CQA model achieves the overall best macro and micro F1 values, achieving state-of-the-art results of 66.25% and 77.71%, respectively. MRL-CQA also achieves the best or second-best performance in six out of the seven categories. Of the three hardest categories (the last three types in Table 1), MRL-CQA delivers the best performance in all three types. This validates the effectiveness of our meta-learning-based approach in

Table 1: Performance comparison (measured in F1) of the seven methods on the CQA test set. For each category, best result is **bolded** and second-best result is underlined.

Question category	KVmem	NSM	CIP-All	CIP-Sep	PG	MRL-CQA
Simple Question	41.40%	88.83%	41.62%	**94.89%**	85.20%	88.37%
Logical Reasoning	37.56%	80.21%	21.31%	**85.33%**	78.23%	80.27%
Quantitative Reasoning	0.89%	36.68%	5.65%	33.27%	44.22%	**45.06%**
Verification (Boolean)	27.28%	58.06%	30.86%	61.39%	84.42%	**85.62%**
Comparative Reasoning	1.63%	59.45%	1.67%	9.60%	59.43%	**62.09%**
Quantitative (Count)	17.80%	58.14%	37.23%	48.40%	61.80%	**62.00%**
Comparative (Count)	9.60%	32.50%	0.36%	0.99%	38.53%	**40.33%**
Overall macro F1	19.45%	59.12%	19.82%	47.70%	64.55%	**66.25%**
Overall micro F1	31.18%	74.68%	31.52%	73.31%	75.40%	**77.71%**

effectively learning task-specific knowledge. Note that the two categories that MRL-CQA performs the best, Comparative Reasoning and Comparative (Count), both account for less than 5% of the training set, which further demonstrates our model's excellent adaptability.

Also, our RL-based programmer PG achieves second-best result in overall macro and micro F1, with about 2% difference below MRL-CQA. Moreover, PG achieves second-best in four categories. Such strong performance indicates the effectiveness of our CQA framework.

Besides the above main result, several important observations can be made from Table 1.

1. CIP-Sep got the best result in two categories, i.e., "Simple Question" and "Logical Reasoning". However, it performed poorly for the three hard categories. Consequently, the overall macro F1 value of CIP-Sep is substantially lower than both PG and MRL-CQA. Note that CIP-Sep trained a different model separately for each of the seven question categories. The results reported for each category were obtained from the models tuned specifically for each category (Saha et al., 2019), which necessitated a classifier to be trained first to recognize the question categories. Thus, CIP-Sep needs to re-train the models to adapt to new/changed categories, which impedes it from generalizing to unseen instances. However, we tuned our models on all questions with one set of model parameters, disregarding the question category information.

2. As presented in Table 1, CIP-All, the model that trained over all types of the questions, performed much worse in all the categories than CIP-Sep. A possible reason for CIP-All's significant performance degradation is that it is hard for such a one-size-fits-all model to find the weights that fit the training data when the examples vary widely. Besides, the imbalanced classes of questions also deteriorate the performance of the model. Different from CIPITR, MRL-CQA is designed to adapt appropriately to various categories of questions with one model, thus only needs to be trained once.

3. Our programmer and carefully-defined primitive actions presented in this work were used in our re-implementation of NSM. In the hard categories, by comparing the F1 scores of PG and MRL-CQA, it could be observed that NSM performed competitively. Furthermore, NSM performed the second best in "Simple Question" and "Comparative Reasoning" categories. This helps to validate the effectiveness of our proposed techniques. However, NSM is worse than MRL-CQA in six out of the seven categories. This verifies the adaptability of our model, which can quickly adapt to new tasks by employing the learned task-specific knowledge.

Note that our model was trained only on 1% of the training set, whereas the baseline models use the entire training set. Besides, our method trains one model to solve all questions, while CIP-Sep trains seven models, one for each category of problems. Thus our model is compared with seven individually trained models in CIPITR but still achieves the best overall performance, demonstrating the effectiveness of our technique.

3.2 Model Analysis

We conduct an ablation experiment to study the effect of meta-learning. We also study the effect of smaller training samples by comparing MRL-CQA's performance trained on 500 and 1K samples, against 2K used in the full model.

Table 2: Ablation study on the test set on macro F1 score change with different sizes of training samples.

Feature	Overall macro F1
PG	75.40%
MRL-CQA	
500-training	+0.01%
1,000-training	+0.58%
2,000-training	+2.31%

Table 2 summarizes the ablation study result. Trained on 500 samples only, MRL-CQA slightly improves performance by 0.01 percentage points compared to PG. When training sample increases to 1K, MRL-CQA outperforms PG by 0.58 percentage points. The full MRL-CQA model, trained on 2K samples, achieves a performance improvement over PG of 2.31 percentage points. These results demonstrate the ability to design a specific model for answering each question precisely, which is afforded by meta-learning.

3.3 Case Study

We provide a case study of different types of questions that MRL-CQA could answer, but our RL-based model, aka PG, could not solve. The comparison is given in Table 3, which lists the action sequences and the corresponding results these two models predicted when answering the same questions. We highlight the different parts of the action sequences that the two models generated.

For example, when answering the Logical Reasoning question in Table 3, PG was confused about what relations should be used to form feasible actions. It could be seen that PG failed to distinguish the two different relations for the two actions and thus produced a wrong answer.

Similarly, when answering the Verification question in Table 3, PG also yielded an infeasible action sequence. After forming a set of political territories that Hermine Mospointner is a citizen of, the bool action should be used to judge whether Valdeobispo and Austria are within the set. It can be seen that PG missed one action: *Bool (Austria)*.

The different optimization goals lead to the different results of the two models. PG, as a typical one-size-fits-all model, aims to estimate the globally optimal parameters by fitting itself to the training samples. Such a model extracts the information from the training data to update model parameters, applies the parameters to the new samples without modification thereafter. Therefore, when facing a wide variety of questions, it is hard for the model to find a set of parameters that fits all samples. Under the circumstances, like what is presented in Table 3, such a one-size-fits-all model could not handle the questions well.

However, our MRL-CQA model aims to learn general knowledge across tasks and fix the knowledge into the initial parameters. We thus learn a model that can subsequently adapt the initial parameters to each new given question and specialize the adapted parameters to the particular domain of the new questions. Therefore, with the help of the adapted parameters, MRL-CQA can answer each new question more precisely than PG.

4 Related Work

Imitation Learning. Imitation Learning aims to learn the policy based on the expert's demonstration by supervised learning. Saha et al. (2018) propose a CQA model that combines Hierarchical Recurrent Encoder-Decoder (HRED) with a Key-Value memory (KVmem) network and predicts the answer by attending on the stored memory. Guo et al. (2018) present a Dialog-to-Action (D2A) approach to answer complex questions by learning from the annotated programs. D2A employs a deterministic BFS procedure to label questions with pseudo-gold actions and trains an encoder-decoder model to generate programs by managing dialog memory. Multi-task Semantic Parsing (MaSP) (Shen et al., 2019) jointly optimizes two modules to solve the CQA task, i.e., entity linker and semantic parser, relying on annotations to demonstrate the desired behaviors. Different from the above approaches, our model performs better while removing the need for annotations.

Neural Program Induction (NPI). NPI is a paradigm for mapping questions into executable programs by employing neural models. Neural-Symbolic Machines (NSM) (Liang et al., 2017) is proposed to answer the multi-hop questions. NSM annotates the questions and then anchors the model to the high-reward programs by assigning them with a deterministic probability. Neural-Symbolic Complex Question Answering (NS-CQA) model (Hua et al., 2020) augments the NPI approach with a memory buffer to alleviate the sparse reward and data inefficiency problems appear in the CQA task. Complex Imperative

Table 3: A comparison of the action sequences and results that PG (the second column) and MRL-CQA (the third column) yield when answering the same questions.

An Example of Logical Reasoning Question		
Question Information	PG	MRL-CQA
Question: Which occupations are the professions of Sergio Piacentini **or** were a position held by Antoinette Sandbach?	**Action sequence:** *Select (Sergio Piacentini, position_held, occupation)* *Union (Antoinette Sandbach, position_held, occupation)*	**Action sequence:** *Select (Sergio Piacentini, occupation_of, occupation)* *Union (Antoinette Sandbach, position_held, occupation)*
Ground-truth answer: Member of the National Assembly for Wales, association football manager, association football player	**Execution result:** Member of the National Assembly for Wales	**Execution result:** Member of the National Assembly for Wales, association football manager, association football player
An Example of Verification (Boolean) Question		
Question Information	PG	MRL-CQA
Question: Is Hermine Mospointner a civilian of Valdeobispo **and** Austria?	**Action sequence:** *Select (Hermine Mospointner, country_of_citizenship, political territory)* *Bool (Valdeobispo)*	**Action sequence:** *Select (Hermine Mospointner, country_of_citizenship, political territory)* *Bool (Valdeobispo)* *Bool (Austria)*
Ground-truth answer: False and True	**Execution result:** False	**Execution result:** False and True

Program Induction from Terminal Rewards (CIPITR) (Saha et al., 2019) relies on auxiliary awards, KB schema, and inferred answer types for training an NPI model to solve the CQA task. However, CIPITR separately trains one model for each category of questions with a different difficulty level. Compared with the NPI models, our model can flexibly adapt to the question under processing.

Meta-learning. Meta-learning, aka learning-to-learn, aims to make learning a new task more effective based on the inductive biases that are meta-learned in learning similar tasks in the past. Huang et al. (2018) use MAML to learn a Seq2Seq model to convert questions in WikiSQL into SQL queries. More closely related to our work, Guo et al. (2019) propose Sequence-to-Action (S2A) by using MAML to solve CQA problems. They label all the examples in training set with pseudo-gold annotations, then train an encoder-decoder model to retrieve relevant samples and a Seq2Seq based semantic parser to generate actions based on the annotations. Unlike S2A, we introduce a Meta-RL approach, which uses RL to train an NPI model without annotating questions in advance.

5 Conclusion

CQA refers to answering complex natural language questions on a KB. In this paper, we propose a meta-learning method to NPI in CQA, which quickly adapts the programmer to unseen questions to tackle the potential distributional bias in questions. We take a meta-reinforcement learning approach to effectively adapt the meta-learned programmer to new questions based on the most similar questions retrieved. To effectively create the support sets, we propose an unsupervised retriever to find the questions that are structurally and semantically similar to the new questions from the training dataset. When evaluated on the large-scale complex question answering dataset, CQA, our proposed approach achieves state-of-the-art performance with overall macro and micro F1 score of 66.25% and 77.71%, respectively.

In the future, we plan to improve MRL-CQA by designing a retriever that could be optimized jointly with the programmer under the meta-learning paradigm, instead of manually pre-defining a static relevance function. Other potential directions of research could be toward learning to cluster questions into fine-grained groups and assign each group a set of specific initial parameters, making the model finetune the parameters more precisely.

Acknowledgments

This work was partially supported by the National Key Research and Development Program of China under grants (2018YFC0830200), the Natural Science Foundation of China grants (U1736204, 61602259), Australian Research Council (DP190100006), the Judicial Big Data Research Centre, School of Law at Southeast University, and the project no. 31511120201 and 31510040201.

References

Ghulam Ahmed Ansari, Amrita Saha, Vishwajeet Kumar, Mohan Bhambhani, Karthik Sankaranarayanan, and Soumen Chakrabarti. 2019. Neural program induction for kbqa without gold programs or query annotations. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 4890–4896. AAAI Press.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org.

Daya Guo, Duyu Tang, Nan Duan, Ming Zhou, and Jian Yin. 2018. Dialog-to-action: conversational question answering over a large-scale knowledge base. In *Advances in Neural Information Processing Systems*, pages 2942–2951.

Daya Guo, Duyu Tang, Nan Duan, Ming Zhou, and Jian Yin. 2019. Coupling retrieval and meta-learning for context-dependent semantic parsing. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 855–866. Association for Computational Linguistics.

Kelvin Guu, Panupong Pasupat, Evan Liu, and Percy Liang. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1051–1062.

Yuncheng Hua, Yuan-Fang Li, Guilin Qi, Wei Wu, Jingyao Zhang, and Daiqing Qi. 2020. Less is more: Data-efficient complex question answering over knowledge bases. *Journal of Web Semantics*, Accepted.

Po-Sen Huang, Chenglong Wang, Rishabh Singh, Wen-tau Yih, and Xiaodong He. 2018. Natural language to structured query generation via meta-learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 732–738.

Hailong Jin, Chengjiang Li, Jing Zhang, Lei Hou, Juanzi Li, and Peng Zhang. 2019. XLORE2: large-scale cross-lingual knowledge graph construction and application. *Data Intell.*, 1(1):77–98.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 23–33.

Arvind Neelakantan, Quoc V. Le, and Ilya Sutskever. 2016. Neural programmer: Inducing latent programs with gradient descent. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Alex Nichol and John Schulman. 2018. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2.

Panupong Pasupat and Percy Liang. 2016. Inferring logical forms from denotations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Amrita Saha, Ghulam Ahmed Ansari, Abhishek Laddha, Karthik Sankaranarayanan, and Soumen Chakrabarti. 2019. Complex program induction for querying knowledge bases in the absence of gold programs. *Transactions of the Association for Computational Linguistics*, 7:185–200.

Amrita Saha, Vardaan Pahuja, Mitesh M Khapra, Karthik Sankaranarayanan, and Sarath Chandar. 2018. Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

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

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Denny Vrandecic and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57:78–85.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 643–648.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206.

Wenpeng Yin, Mo Yu, Bing Xiang, Bowen Zhou, and Hinrich Schütze. 2016. Simple question answering by attentive convolutional neural network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1746–1756.

Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2017. Improved neural relation detection for knowledge base question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 571–581.

Chapter 5

Complex Knowledge Base Question Answering via Alternate Meta-learning

In Chapter 4, instead of using a monolithic, one-size-fits-all model, we present a novel Meta-RL framework, which adaptively generates multiple models for multiple groups of complex questions to solve the CQA task. The key idea underlying a meta-learning algorithm is to train a model that could quickly adapt to a new task by learning from a small number of samples that belong to this new task [179]. Unlike other few-shot learning tasks such as the few-shot image recognition tasks on the Omniglot [190] and MiniImagenet [178] datasets, in the semantic parsing setting, such a notion of *"support set"*, which denotes a group of similar samples representing each novel *task*, is not provided in the dataset. Therefore, the meta-learning framework is faced with a challenge when applying to semantic parsing problems, including the CQA problem: the support sets are not available in the given dataset, making it impossible to train a meta-learner [17]. To address the above challenge in the CQA task, the notion of *retriever* is introduced to construct support sets to conduct meta-learning. In meta-learning process, each training question in the dataset is viewed as a *"test sample"* of a unique *pseudo-task*, and the K samples which are most relevant to this *test sample*—that are selected from the training corpus by the *retriever* with the help of a *relevance function*—are treated as *"training samples"*, aka *support set*, for this specific *pseudo-task* [17, 191]. The objective of meta-learning is then to deliver a model that could minimize the loss incurred on the *test sample* of a *pseudo-task* by training with the corresponding *support set*. Therefore, the retriever's design, which is used to form the *support sets*, is crucial for a meta-learning framework.

Recently, numerous prior works have designed *non-learned* retrievers to handle support set collection problems in semantic parsing tasks. Huang et al. [17] train a classifier to predict the question's type by using the bag-of-words features of the questions. Then they pick the samples belonging to the same type as the *training samples* while having the same length. Unlike this work, Guo et al. [18] learn a retriever independent of manual-defined relevance function. Guo et al. first employ a Brute-Force search algorithm to pair the given training questions with corresponding *pseudo-gold annotations*, and then utilize the question-annotation pairs to train an encoder-decoder model, where a question is encoded into a latent vector that could produce the relevant annotation by the decoder. Then they could retrieve the samples within a short distance from the *test sample* in the latent vector space. Both approaches rely on annotated labels for learning a retriever, making them inapplicable to the settings where the annotations are inaccessible. In contrast, as depicted in Chapter 4, we formulate the retriever under an unsupervised learning setup avoiding the necessity of annotations. We propose an unsupervised relevance function for computing the similarity between the candidate training samples and the *test sample*. We measure the similarity in two aspects: (i) structural similarity by comparing the number of KB artifacts and (ii) lexical similarity using neural word embedding.

Though these above approaches can have the desirable effect of retrieving relevant samples, the retrievers are *non-learned*, i.e., fixed when conduct meta-learning. The retrievers would not update in training the meta-learner, being trained independently of the meta-learner. Therefore, it is hard to investigate the effect of support set construction for the problem of meta-learning and thus difficult to measure the retriever's performance. Instead, we design a novel meta-learning framework that the retriever could be trained along with the meta-learner. The result of the meta-learner answering the *test sample* could be used as a weak supervision signal for training the retriever.

This chapter investigates the role that support-set construction plays in meta-learning and provides some insights into retriever training. We empirically demonstrate the performance improvement of the high-quality support sets constructed with our learned retriever. Particularly, we design a retriever that can be trained:

1. jointly with the meta-learner and
2. using weak supervisions instead of annotations.

Therefore, we extend the Meta-RL framework described in Chapter 4 by proposing a novel *retriever*. We name it as **M**et**A** **R**etrieval **L**earning (**MARL**). This research work has been published

in the *Twenty-Ninth International Joint Conference on Artificial Intelligence and Seventeenth Pacific Rim International Conference on Artificial Intelligence, IJCAI-PRICAI-20.* The complete version of the manuscript is attached in the subsequent pages.

Hua, Y., Li, Y., Haffari, G., Qi, G. and Wu, W. Retrieve, Program, Repeat: Complex Knowledge Base Question Answering via Alternate Meta-learning. In *Twenty-Ninth International Joint Conference on Artificial Intelligence and Seventeenth Pacific Rim International Conference on Artificial Intelligence, IJCAI-PRICAI-20*, Yokohama, Japan, January 7-15, 2021, Proceedings, pages 3679-3686.

Retrieve, Program, Repeat: Complex Knowledge Base Question Answering via Alternate Meta-learning

Yuncheng Hua[1,3], **Yuan-Fang Li**[2], **Gholamreza Haffari**[2], **Guilin Qi**[1,4*] and **Wei Wu**[1]
[1]School of Computer Science and Engineering, Southeast University, Nanjing, China
[2]Faculty of Information Technology, Monash University, Melbourne, Australia
[3]Southeast University-Monash University Joint Research Institute, Suzhou, China
[4]Key Laboratory of Computer Network and Information Integration, Southeast University, China
{devinhua, gqi, wuwei}@seu.edu.cn, {yuanfang.li, gholamreza.haffari}@monash.edu

Abstract

A compelling approach to complex question answering is to convert the question to a sequence of actions, which can then be executed on the knowledge base to yield the answer, aka the programmer-interpreter approach. Use similar training questions to the test question, meta-learning enables the programmer to adapt to unseen questions to tackle potential distributional biases quickly. However, this comes at the cost of manually labeling similar questions to learn a retrieval model, which is tedious and expensive. In this paper, we present a novel method that automatically learns a retrieval model alternately with the programmer from weak supervision, i.e., the system's performance with respect to the produced answers. To the best of our knowledge, this is the first attempt to train the retrieval model with the programmer jointly. Our system leads to state-of-the-art performance on a large-scale task for complex question answering over knowledge bases. We have released our code at https://github.com/DevinJake/MARL.

1 Introduction

Complex question answering (CQA) over knowledge base (KB) aims to map natural-language questions to logical forms (*annotations*), i.e., programs or action sequences, which can be directly executed on the KB to yield answers (*denotations*) [Berant *et al.*, 2013; Shen *et al.*, 2019]. Different from other forms of KBQA, such as multi-hop question answering, CQA requires discrete *aggregation* actions, such as set intersection/union, counting, and min/max, to yield answers, which can be entities in the KB as well as numbers.

Taking one dataset CQA [Saha *et al.*, 2018] as an example, the questions are organized into seven categories, and the questions in the different categories vary substantially in length and complexity. For instance, as shown in Figure 1, the question in the 'Simple' category only requires one 'select' action to answer, while the question in the 'Quantitative Count' category requires the execution of the sequence of actions 'select', 'intersection' and 'count' to obtain the answer.

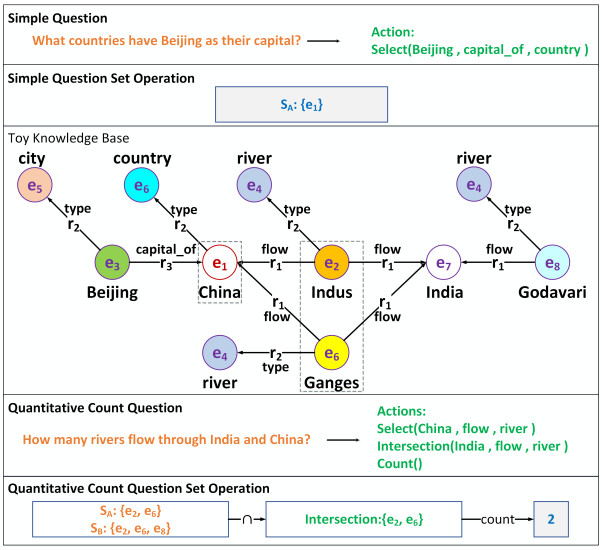

Figure 1: Two questions of different types in the CQA dataset.

Standard KBQA approaches [Berant *et al.*, 2013; Yih *et al.*, 2014; Bordes *et al.*, 2015; Yu *et al.*, 2017; Guo *et al.*, 2018; Saha *et al.*, 2018; Ansari *et al.*, 2019] adopt imitation learning, memory network, or RL, and they typically train one model that fits the entire training dataset and use it to answer all questions. Such a one-size-fits-all model aims to learn the generic knowledge across the questions in the training phase and use the knowledge to predict action sequences for each test question in the inference phase. However, when the training questions vary widely, less common knowledge will be shared across them. Thus the increase in the diversity of the questions is associated with a reduction in the generic knowledge. It would be challenging for the one-size-fits-all model to produce optimal logical forms for each instance.

Several methods have recently been proposed to address this challenge. Complex Imperative Program Induction from Terminal Rewards (CIPITR) [Saha *et al.*, 2019] takes a neural program induction (NPI) approach and proposes to train different independent models, one for each category of CQA questions. Essentially, CIPITR aims to learn a one-size-fits-all model for each *question type* instead of granting the model

*Contact Author

the ability to transfer from one question type to another.

S2A [Guo *et al.*, 2019] attempts to address this challenge with a method based on a meta-learning method [Schmidhuber, 1992], employing Model Agnostic Meta-Learning (MAML) [Finn *et al.*, 2017] specifically. S2A puts forward a retriever and a meta-learner. It first collects annotations for each question in the training set and trains a retriever to find questions with similar annotations. Subsequently, in training the meta-learner, when faced with a new question, the already-trained retriever finds samples from the training set that are similar to the new question and regards these samples as a support set. The meta-learner views the new question along with the support set as a new task and then learns a specific model adaptive to the task.

However, S2A is faced with two difficulties. Firstly, as it employs a teacher forcing approach in the process of training the retriever, it places a burden on collecting annotations for each question. Secondly, the retriever is trained independently of the meta-learner. Thus, the result of the meta-learner answering the questions is irrelevant to the retriever. Therefore it is hard to evaluate the quality of the support set that the retriever establishes for each new question and consequently tricky to measure the impact of the retriever on answering the questions. If the samples found by the retriever are not similar to the new question as expected, the meta-learner will be misguided by the deviated examples and thus learns a model that is not suitable for the current task.

Furthermore, though the approach taken by S2A is the most similar to ours, the tasks differ significantly. S2A aims to answer *context-dependent* questions, where each question is part of a multiple-round conversation. Hence, the context-aware retriever proposed in S2A considers the relevant conversation. On the contrary, however, in this paper, we consider the setup where the questions are single-round and directly answerable from the KB. Thus, a novel challenge arises in retrieving accurate support sets without conversation-based context information.

In this work, to address the above problems, we propose MetA Retrieval Learning (MARL), a new learning algorithm that jointly optimizes retriever and programmer in two stages.

In the first stage, we fix the parameter of the retriever and employ it to select the top-N similar questions (secondary questions) to a given target question (primary question). The *trial* trajectories, along with the corresponding rewards for answering each secondary question, are used to adapt the programmer to the current primary question. The feedback on how correctly the adaptive programmer answers the primary question is used to update the weights of the programmer.

In the second stage, we fix the parameter of the programmer and optimize the retriever. The *general* programmer first outputs an answer and gain a reward to the primary question without using any secondary questions. Then we sample M different sets of secondary questions following the retriever policy and employ the question sets to learn M different programmers. Each specific programmer will generate an answer to the primary question and gain some reward. We consider the difference between the reward yielded by the general programmer and the reward gained by each adapted programmer as the contribution of employing the corresponding support set. Thus, the reward difference provides the training signal to optimize the policy of the retriever: positive difference increases the probability that a set of secondary questions is chosen, and a negative difference lowers the probability.

We train the programmer and the retriever alternatively until the models converge. Note that in our method, the training of the retriever is done in a weakly-supervised manner. The retriever is optimized to find better support sets according to the programmer's performance of answering primary questions, rather than employing teacher forcing. Since one support set generates one adaptive programmer, we employ the feedback obtained by evaluating the adaptive programmer as a weak supervision signal to optimize the retriever. Therefore we encourage the retriever to find the support set that leads to a superior programmer that gains more reward. At the same time, the programmer is optimized alongside the retriever.

We evaluate our method on CQA [Saha *et al.*, 2018], a large-scale complex question answering dataset. MARL outperforms standard imitation learning or RL-based methods and achieves new state-of-the-art results with overall micro and macro F1 score. Notably, MARL uses only 1% of the training data to form the tasks for meta-learning and achieve competitive results.

2 Alternate Meta-learning for Complex Question Answering over Knowledge Bases

We now introduce our method for combining weakly-supervised *retriever* and *meta-learning* to solve the CQA task, which we call MARL.

In this paper, we consider each new complex question in the training dataset as one *primary question* and view answering one primary question as an individual task. To solve an unseen task effectively, we aim to build a unique *programmer* for each task. Thus we harness the *retriever* to find the top-N most similar questions as the *secondary questions*, and propose a meta reinforcement learning approach to rapidly adapt the programmer to the primary question with the support of the secondary questions.

2.1 Method Overview

The goal of MARL is to train the retriever to find the optimal secondary questions for the programmer, which, when faced with a new primary question, can quickly adapt the programmer to the unseen question and effectively improve QA performance. To accomplish this goal, we train two networks jointly: (1) an encoder-decoder network, which is viewed as the *programmer*, that transforms questions into programs; and (2) a *retriever* network, which finds the secondary questions, i.e., the N questions that are the most analogous to the primary question.

We denote the encoder-decoder network as a policy $\pi(\boldsymbol{\tau}|\boldsymbol{q}_i;\boldsymbol{\theta})$ with parameter $\boldsymbol{\theta}$, and the retriever network as a policy $\pi(\boldsymbol{q}_{c_i}|\boldsymbol{q}_{pri};\boldsymbol{\phi})$ with parameter $\boldsymbol{\phi}$. Both networks take questions as the input, while the output of the programmer network is the programs that can be directly executed on the KB to generate answers, and the production of the retriever network is the probability distribution over all candidate training samples. For the retriever network, the N sam-

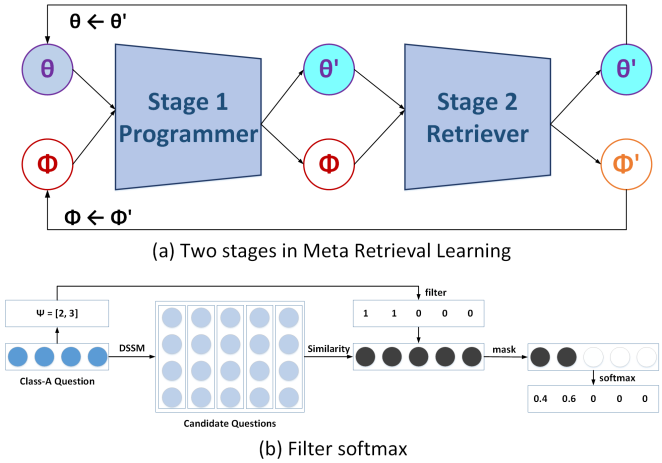

Figure 2: (a). Illustration of the two stages in the training process of MARL. (b) Illustration of the filter softmax.

ples that have the highest probabilities are selected as secondary questions for a given primary question. Both the parameters $\boldsymbol{\theta}$ and ϕ are optimized by the rewards, which are the comparison between the generated answers and the ground-truth answers.

In the training process, when the primary question and the corresponding secondary questions are similar (i.e., with analogical structures and semantic meanings), the programmer will be guided more effectively in finding the optimal direction of parameter update. On the contrary, if the secondary questions are randomly extracted from the training set, they might share a little commonality with the primary question and consequently yield disparate action sequences. Since only a small amount of samples are used to finetune the adaptive model, such dissimilar instances will lead the programmer on a noisier (more random) path for gradient update.

In the retriever network, we thus cluster the training samples into several groups based on the question type and use a vector ψ to record group membership of each question. For each primary question, based on ψ, a unique *filter* $\boldsymbol{F}$ will be generated to determine the possible questions as candidate secondary questions by filtering out irrelevant questions. Given the primary question $\boldsymbol{q}_{pri}$, the retriever network computes the probability of choosing a candidate question $\boldsymbol{q}_{c_i}$ as a secondary question by applying a filter softmax function, which could be denoted as

$$\pi(\boldsymbol{q}_{c_i}|\boldsymbol{q}_{pri}; \psi, \phi) \tag{1}$$

Following the probability, we sample a set of secondary questions and denote them as $s^{\boldsymbol{q}_{pri}}$, which is used to support adapting the programmer for the primary question $\boldsymbol{q}_{pri}$.

2.2 Model Objectives

We consider two types of knowledge that can contribute to the answering of a new question: the task-specific knowledge, which is the particular features shared with a small number of similar questions that inform what the current task is; and the task-agnostic knowledge, which is the generic features shared across different tasks. If similar questions are accurately selected, the model could effectively exploit the particular features among these questions and acquire task-specific knowledge to recognize the new task. If the generic features are suitable for many tasks, the model could rapidly adapt itself to a new task and produce satisfactory results for the task by conducting only a small number of gradient updates. We employ a retriever to find similar questions for acquiring the task-specific knowledge and train a generic policy to accumulate task-agnostic knowledge.

In our method, the performance of answering a new question is represented by the reward feedback on whether the question is correctly answered. It is hard to disentangle the contribution of the two types of knowledge. In other words, it is difficult to determine which factor is the reason for generating a correct answer, the task-specific knowledge gained from similar questions, or the task-agnostic knowledge acquired from the well-learned generic policy.

As shown in Figure 2(a), we split the training process in each epoch into two independent stages and train the programmer and the retriever alternately. In the **first stage**, we fix the parameter ϕ of the retriever network and employ it to select secondary questions to optimize the parameter $\boldsymbol{\theta}$ of the programmer. The target of the first stage is to encourage the model to learn the generic knowledge that is broadly applicable to all tasks by updating $\boldsymbol{\theta}$. In the **second stage**, we fix the parameter $\boldsymbol{\theta}$ of the programmer and update the retriever network by computing its gradients with respect to the programmer's output. We compare the results of answering the same primary question with or without using the retriever and consider the difference between the results as the indication of how the retriever contributes to solving the primary question. Such differences are used to train the retriever to find the optimal set of secondary questions (the *support set*). Both networks are trained iteratively until convergence.

The training approach is depicted in Algorithm 1.

In the **first stage** (lines 4–12 in Algorithm 1) of each epoch, we propose a *meta reinforcement learning* (meta-RL) approach to update $\boldsymbol{\theta}$, i.e. the programmer. We use the gradient-based meta-learning method to solve the meta-RL problem such that we can obtain the optimal policy for a given task after performing a few steps of vanilla policy gradient (VPG) with the task. We employ *Monte Carlo integration* (MC) as the approximation strategy in the Policy Gradient.

The meta-learning process is divided into two steps to solve a task, namely the meta-training step and the meta-test step. Suppose we are trying to answer the primary question $\boldsymbol{q}_{pri}$, N secondary questions $s^{\boldsymbol{q}_{pri}}$ will be first found by the retriever network, and we consider $\boldsymbol{q}_{pri}$ together with $s^{\boldsymbol{q}_{pri}}$ as a pseudo-task $\mathcal{T}_{pse}$. During meta-training, the meta-RL model first generates K trajectories for each question in $s^{\boldsymbol{q}_{pri}}$ based on parameter $\boldsymbol{\theta}$. The reward of each trajectory is given by the environment and subsequently used to adapt $\boldsymbol{\theta}'$ to task $\mathcal{T}_{pse}$ as follows:

$$\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta} + \eta_1 \nabla_{\boldsymbol{\theta}} \sum_{\boldsymbol{q}_i \in s^{\boldsymbol{q}_{pri}}} \mathbb{E}_{\boldsymbol{\tau} \sim \pi(\boldsymbol{\tau}|\boldsymbol{q}_i;\boldsymbol{\theta})}[R(\boldsymbol{\tau})] \tag{2}$$

In the meta-test step, another K trajectories corresponding to the primary question $\boldsymbol{q}_{pri}$ are further produced by $\boldsymbol{\theta}'$. The

reward produced by the new trajectories is considered as the evaluation of the adapted policy $\boldsymbol{\theta}'$ for the given task $\mathcal{T}_{pse}$; thus, we have the following objective:

$$J_{\boldsymbol{q}_{pri}}(\boldsymbol{\theta}') \stackrel{\text{def}}{=} \mathbb{E}_{\boldsymbol{\tau}' \sim \pi(\boldsymbol{\tau}'|\boldsymbol{q}_{pri};\boldsymbol{\theta}')}[R(\boldsymbol{\tau}')] \tag{3}$$

The parameter of the generic policy $\boldsymbol{\theta}$ are then trained by maximizing the objective $J(\boldsymbol{\theta}')$,

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta_2 \nabla_{\boldsymbol{\theta}} J_{\boldsymbol{q}_{pri}}(\boldsymbol{\theta}') \tag{4}$$

In each VPG step, since we have N samples in $\boldsymbol{s}^{\boldsymbol{q}_{pri}}$, we use N policy gradient adaptation steps to update $\boldsymbol{\theta}'$. Meanwhile, we use one policy gradient step to optimize $\boldsymbol{\theta}$ based on the evaluation of $\boldsymbol{\theta}'$. We denote the optimized parameter of the programmer $\boldsymbol{\theta}^*$.

In the **second stage** (lines 13–25 in Algorithm 1) of each epoch, we propose a *reinforcement learning* approach to update $\boldsymbol{\phi}$, i.e. the retriever. The primary questions that we want to solve are the same as the data used in the first stage. When answering the primary question $\boldsymbol{q}_{pri}$, we first generate a trajectory based on parameter $\boldsymbol{\theta}^*$ that has been optimized in the first stage as:

$$\boldsymbol{\tau}^* \leftarrow \text{decode}(\pi(\boldsymbol{\tau}|\boldsymbol{q}_{pri};\boldsymbol{\theta}^*)) \tag{5}$$

and further, obtain the reward $R(\boldsymbol{\tau}^*)$ by executing the trajectory.

We then employ the retriever to compute the probability of selecting one candidate question as a secondary question. From this distribution, we sample N secondary questions to form a support set, rather than directly choosing the N questions with the highest probability. The probability of sampling a set of questions $\boldsymbol{s}^{\boldsymbol{q}_{pri}}$ is:

$$P(\boldsymbol{s}^{\boldsymbol{q}_{pri}}) = \prod_{\boldsymbol{q}_{c_i} \in \boldsymbol{s}^{\boldsymbol{q}_{pri}}} \pi(\boldsymbol{q}_{c_i}|\boldsymbol{q}_{pri},\boldsymbol{\psi};\boldsymbol{\phi}) \tag{6}$$

We could repeat sampling several times and acquire different support sets. Employ one set $\boldsymbol{s}_m^{\boldsymbol{q}_{pri}}$ from these support sets, as what we have done in the first stage, we get the adapted $\boldsymbol{\theta}_m^{*'}$:

$$\boldsymbol{\theta}_m^{*'} \leftarrow \boldsymbol{\theta}^* + \eta_1 \nabla_{\boldsymbol{\theta}^*} \sum_{\boldsymbol{q}_i \in \boldsymbol{s}_m^{\boldsymbol{q}_{pri}}} \mathbb{E}_{\boldsymbol{\tau} \sim \pi(\boldsymbol{\tau}|\boldsymbol{q}_i;\boldsymbol{\theta}^*)}[R(\boldsymbol{\tau})] \tag{7}$$

Then we generate a new trajectory under $\boldsymbol{\theta}_m^{*'}$ for the primary question:

$$\boldsymbol{\tau}_m^{*'} \leftarrow \text{decode}(\pi(\boldsymbol{\tau}|\boldsymbol{q}_{pri};\boldsymbol{\theta}_m^{*'})) \tag{8}$$

and also compute the reward for this trajectory as $R(\boldsymbol{\tau}_m^{*'})$.

We regard the difference between $R(\boldsymbol{\tau}^*)$ and $R(\boldsymbol{\tau}_m^{*'})$ as the contribution of the support set $\boldsymbol{s}_m^{\boldsymbol{q}_{pri}}$, which is used to learn the task-specific knowledge from the questions in $\boldsymbol{s}_m^{\boldsymbol{q}_{pri}}$.

The retriever network is then updated by encouraging the particular support sets to be chosen such that, if the policy $\boldsymbol{\theta}$ is adapted to the current task by using these support sets, the reward of answering the primary question would be maximized. Therefore, we harness the difference as the reward and have the objective:

$$J_{\boldsymbol{q}_{pri}}(\boldsymbol{\phi}) \stackrel{\text{def}}{=} \mathbb{E}_{\boldsymbol{s}_m^{\boldsymbol{q}_{pri}} \sim P(\boldsymbol{s}^{\boldsymbol{q}_{pri}})}[R(\boldsymbol{\tau}_m^{*'}) - R(\boldsymbol{\tau}^*)] \tag{9}$$

Algorithm 1: The MARL algorithm

Input: Training dataset Q_{train}, step size η_1, η_2, η_3
Output: The learned $\boldsymbol{\theta}^*$ and $\boldsymbol{\phi}^*$

1 Initialize groups vector $\boldsymbol{\psi}$
2 Randomly initialize $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$
3 **while** *not converged* **do**
4 **foreach** *training iteration* **do**
5 Sample a batch of primary questions $Q_{pri} \sim Q_{train}$
6 **foreach** $\boldsymbol{q}_{pri} \in Q_{pri}$ **do**
7 Retrieve $\boldsymbol{s}^{\boldsymbol{q}_{pri}}$ with $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$
8 $\mathcal{L} = \sum_{\boldsymbol{q}_i \in \boldsymbol{s}^{\boldsymbol{q}_{pri}}} \mathbb{E}_{\boldsymbol{\tau} \sim \pi(\boldsymbol{\tau}|\boldsymbol{q}_i;\boldsymbol{\theta})}[R(\boldsymbol{\tau})]$
9 Get adapted parameters: $\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta} + \eta_1 \nabla_{\boldsymbol{\theta}} \mathcal{L}$
10 $J_{\boldsymbol{q}_{pri}}(\boldsymbol{\theta}') \stackrel{\text{def}}{=} \mathbb{E}_{\boldsymbol{\tau}' \sim \pi(\boldsymbol{\tau}'|\boldsymbol{q}_{pri};\boldsymbol{\theta}')}[R(\boldsymbol{\tau}')]$
11 Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta_2 \nabla_{\boldsymbol{\theta}} \sum_{\boldsymbol{q}_{pri} \in Q_{pri}} J_{\boldsymbol{q}_{pri}}(\boldsymbol{\theta}')$
12 $\boldsymbol{\theta}^* \leftarrow \boldsymbol{\theta}$
13 **foreach** *training iteration* **do**
14 Sample a batch of primary questions $Q_{pri} \sim Q_{train}$
15 **foreach** $\boldsymbol{q}_{pri} \in Q_{pri}$ **do**
16 $\boldsymbol{\tau}^* \leftarrow \text{decode}(\pi(\boldsymbol{\tau}|\boldsymbol{q}_{pri};\boldsymbol{\theta}^*))$
17 Compute reward $R(\boldsymbol{\tau}^*)$
18 Sample support sets $\mathcal{M}$ with $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$
19 **foreach** $\boldsymbol{s}_m^{\boldsymbol{q}_{pri}} \in \mathcal{M}$ **do**
20 $\boldsymbol{\theta}_m^{*'} \leftarrow \boldsymbol{\theta}^* + \eta_1 \nabla_{\boldsymbol{\theta}^*} \sum_{\boldsymbol{q}_i \in \boldsymbol{s}_m^{\boldsymbol{q}_{pri}}} \mathbb{E}_{\boldsymbol{\tau} \sim \pi(\boldsymbol{\tau}|\boldsymbol{q}_i;\boldsymbol{\theta}^*)} R(\boldsymbol{\tau})$
21 $\boldsymbol{\tau}_m^{*'} \leftarrow \text{decode}(\pi(\boldsymbol{\tau}|\boldsymbol{q}_{pri};\boldsymbol{\theta}_m^{*'}))$
22 Compute reward $R(\boldsymbol{\tau}_m^{*'})$
23 $J_{\boldsymbol{q}_{pri}}(\boldsymbol{\phi}) \stackrel{\text{def}}{=} \mathbb{E}_{\boldsymbol{s}_m^{\boldsymbol{q}_{pri}} \sim P(\boldsymbol{s}^{\boldsymbol{q}_{pri}})}[R(\boldsymbol{\tau}_m^{*'}) - R(\boldsymbol{\tau}^*)]$
24 Update $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} + \eta_3 \nabla_{\boldsymbol{\phi}} \sum_{\boldsymbol{q}_{pri} \in Q_{pri}} J_{\boldsymbol{q}_{pri}}(\boldsymbol{\phi})$
25 $\boldsymbol{\phi}^* \leftarrow \boldsymbol{\phi}$
26 **Return** The learned $\boldsymbol{\theta}^*$ and $\boldsymbol{\phi}^*$

The parameter of the retriever network $\boldsymbol{\phi}$ are then updated by maximizing the objective $J(\boldsymbol{\phi})$ as:

$$\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} + \eta_3 \nabla_{\boldsymbol{\phi}} J(\boldsymbol{\phi}) \tag{10}$$

However, it is often infeasible to compute the gradient in Equation (9) because it involves taking an expectation over all possible sampled support sets. Hence, in practice, we employ Monte Carlo integration to approximate the expectation, which is:

$$\Delta_{MC} = \frac{1}{M} \sum_{\boldsymbol{s}_m^{\boldsymbol{q}_{pri}} \in \mathcal{M}} [R(\boldsymbol{\tau}_m^{*'}) - R(\boldsymbol{\tau}^*) - \mathcal{C}] \nabla log(P(\boldsymbol{s}_m^{\boldsymbol{q}_{pri}})) \tag{11}$$

where $\mathcal{M}$ is a collection of M sets of secondary questions retrieved to support the primary question $\boldsymbol{q}_{pri}$, and $\mathcal{C}$ is a baseline with a constant value to reduce the variance of the estimate without altering its expectation.

2.3 Filter Softmax

As mentioned before, we categorize the instances in the training set based on the question type and employ a vector $\boldsymbol{\psi}$ to record the type information of the questions. For each question type, based on $\boldsymbol{\psi}$, a filter $\boldsymbol{F}$ will be created to make the retriever only consider questions of the same type when searching for secondary questions for a primary question of that type. The filter value is set to zero for all instances that do not have the same type of a given question [Liu *et al.*, 2019]. For example, as shown in Figure 2(b), assume we have a dataset with two question types $y = [A, B]$, and we arrange the dataset to group the same type questions together. Thus we have a $\boldsymbol{\psi} = [2, 3]$ to indicate that two questions belong to type A and three questions from type B. In this case, the filter is set as $\boldsymbol{F} = [1, 1, 0, 0, 0]$ for the primary question of type A and $\boldsymbol{F} = [0, 0, 1, 1, 1]$ for type B. Applying the filter to the softmax function would mask the irrelevant questions, which have types different from the primary question. This allows the retriever to reduce the search space and increase the probability of finding expected questions. The filter softmax function produces the following probability:

$$p(q_{c_i}|q_{pri}) = \frac{e^{sim(q_{pri},q_{c_i})} \odot \boldsymbol{F}_i}{\sum_{i'}(e^{sim(q_{pri},q_{c_i'})} \odot \boldsymbol{F}_{i'})}, \qquad (12)$$

where $p(q_{c_i})$ represents the probability that the candidate question q_{c_i} is selected as one of the secondary questions for the primary question q_{pri}, and $sim(q_{pri}, q_{c_i})$ is the semantic similarity between them. Besides, $\boldsymbol{F}_i$ is the binary value in the filter $\boldsymbol{F}$ that recognizes whether question q_{c_i} has the same type as q_{pri} or not, and $\odot$ represents element-wise multiplication. In our work, Deep Structured Semantic Model (DSSM) method [Huang *et al.*, 2013] is employed to compute the similarity between two questions.

3 Evaluation

We evaluated our MARL model on the CQA dataset [Saha *et al.*, 2018]. CQA is a large-scale complex question answering dataset containing 944K/100K/156K question-answer pairs for training/validation/test. CQA divides itself into seven categories based on the nature of answers, e.g., entities as answers in the 'Simple Question' category and numbers in the category 'Quantitative (Count)'. We used 'accuracy' as the evaluation metric for questions whose type is 'Verification', 'Quantitative (Count)', and 'Comparative (Count)'; and 'F1 measure' to other kinds of questions. However, to simplify the presentation and stay consistent with literature [Saha *et al.*, 2019; Ansari *et al.*, 2019], we denote 'accuracy' as 'F1 measure' in Table 1. Hence, the model performance is evaluated on the F1 measure in this paper. Furthermore, we compute the micro F1 and macro F1 scores for all the models based on the F1-scores of the seven question types.

Also, in our analysis of the CQA dataset, we found that the seven types of questions vary substantially in complexity. We discovered that 'Simple' is the simplest that only requires two actions to answer a question, whereas 'Logical Reasoning' is more difficult that requires three actions. Categories 'Verification' and 'Quantitative Reasoning' are the next in the order of difficulty, which need 3–4 actions to answer. The most difficult categories are 'Comparative Reasoning', 'Quantitative (Count)', and 'Comparative (Count)', needing 4–5 actions to yield an answer. Saha et al. [2019] drew a similar conclusion in the manual inspection of these seven question categories.

3.1 Implementation Details

In the CQA dataset, we randomly sampled approximately 1% of the training set (10,353 out of 944K training samples) and annotated them with pseudo-gold action sequences with a breadth-first-search (BFS) algorithm [Guo *et al.*, 2018]. We denote this dataset as Q_{pre}. We trained a BiLSTM-based programmer with Q_{pre}, and further optimized it through RL with another 1% unannotated questions from the training set. We note this RL-based model is a one-size-fits-all model and denote it as **Vanilla**. We randomly selected another 2,072 samples from the 944K training questions to establish pseudo-tasks for meta-learning, which represented only approx. 0.2% of the training set. This model that jointly learns the programmer along with the retriever by employing MAML is our full model and is denoted **MARL**.

We implemented the MARL model in PyTorch with all the weights initialized randomly. We randomly initialized the word embeddings to represent the tokens in questions and output action sequences. We updated the word embeddings within the process of training the Vanilla model and fixed them when training our MARL model. Therefore the primary and the candidate questions in the training dataset were represented as the sum of the vector for each token. The DSSM model took such representation of the questions as the input to compute the semantic similarity between the questions.

In the programmer learning stage, we employed Reptile [Nichol and Schulman, 2018] for fast and simple implementation of MAML while avoiding the significant expense of computing second-order derivatives. We set $\eta_1 = 1e\text{-}4$ when adapting the model to each new task, and set $\eta_2 = 0.1$ to optimize $\boldsymbol{\theta}$ with the gradient update derived from the meta-test data. The reward that the adaptive programmer gained was used to update the retriever parameter $\boldsymbol{\phi}$ through the AdaBound optimizer [Luo *et al.*, 2019] in which the learning rate η_3 was initially set to $1e\text{-}3$ and the final (SGD) learning rate was set to 0.1.

When finding the top-N support set, we set $N = 5$. For each question, we generated five action sequences to output the answers and rewards. Adam optimizer was applied in RL to maximize the expected reward.

In the retriever learning stage, we employed the REINFORCE model to optimize the non-differentiable objective directly. The baseline $\mathcal{C}$ used in REINFORCE was set with a constant value of $1e\text{-}3$ to reduce the variance. We sampled $M = 5$ different sets of the secondary questions and got one unique adaptive programmer for each set with the same meta-learning configuration in the programmer learning stage.

As solving the entity linking problem is beyond the scope of this work, we separately trained an entity/class/relation linker, achieving an accuracy of 99.97%, 91.93%, and 94.29%, respectively. When training the MARL model, the predicted entity/class/relation annotations, along with natural language questions, were used as the input sequence.

Question category	KVmem	CIPITR-All	CIPITR-Sep	Vanilla	Random	Jaccard	MARL
Simple Question (462K)	41.40%	41.62%	**94.89%**	84.67%	85.22%	86.07%	88.06%
Logical Reasoning (93K)	37.56%	21.31%	**85.33%**	76.58%	78.87%	78.89%	79.43%
Quantitative Reasoning (99K)	0.89%	5.65%	33.27%	47.20%	48.11%	48.21%	**49.93%**
Verification (Boolean) (43K)	27.28%	30.86%	61.39%	81.94%	85.04%	85.24%	**85.83%**
Comparative Reasoning (41K)	1.63%	1.67%	9.60%	58.05%	61.96%	63.07%	**64.10%**
Quantitative (Count) (122K)	17.80%	37.23%	48.40%	60.36%	60.04%	60.47%	**60.89%**
Comparative (Count) (42K)	9.60%	0.36%	0.99%	39.25%	38.50%	39.50%	**40.50%**
Overall macro F1	19.45%	19.82%	47.70%	64.01%	65.39%	65.92%	**66.96%**
Overall micro F1	31.18%	31.52%	73.31%	74.72%	75.69%	76.31%	**77.71%**

Table 1: Performance comparison (measured in F1) on the CQA test set. For each category, best result is **bolded** and second-best result is underlined. The number of instances in each category in the training set is also given next to the category name.

Feature	Overall micro F1
Vanilla	74.72%
MARL (random retriever)	+0.97%
MARL (Jaccard retriever)	+1.59%
MARL (full model)	+2.99%

Table 2: Ablation study on the test set on macro F1 score change with the addition of meta-learning and different retrievers. Full model (MARL) has micro F1 of 77.71% as shown in Table 1.

We trained the MARL model with the batch size of 1 and stopped training when the accuracy on the validation set converged (at around 30 epochs). We release the source code at https://github.com/DevinJake/MARL to facilitate replication.

3.2 Performance Evaluation

We compared our model with two baseline methods on the CQA task: KVmem [Saha *et al.*, 2018] and CIPITR [Saha *et al.*, 2019]. Saha et al. [2018] propose KVmem, a baseline CQA model that combines Hierarchical Recurrent Encoder-Decoder (HRED) with a Key-Value memory (KVmem) network. The KVmem model retrieves the most related memory to predict the answer from candidate words by attending on the encoded vectors stored in the memory.

CIPITR [Saha *et al.*, 2019] takes a further step that employs the NPI approach to solving the CQA task without annotations. CIPITR designs high-level constraints to guide the programmer to produce semantically plausible programs for a question. It is worth noting that CIPITR separately trains *a separate model for each of the seven question categories*, and selects the corresponding model to answer questions of the relevant type. We denote the model learned in this way as **CIPITR-Sep**. Besides, CIPITR also trains *one single model over all types of training examples* and uses this single model to answer all questions. We denote this single model as **CIPITR-All**.

Also, we compared our full model, MARL, with several model variants to understand the effect of our retriever and meta-learner. Specifically, **Vanilla** is a BiLSTM-based model further optimized with reinforcement learning. Both Random and Jaccard are MAML-based models with different retrievers. **Random** denotes the model with a retriever that randomly selects questions within the same category. **Jaccard** means the model with a non-learning retriever that makes use of Jaccard similarity on question words.

We ran the open-source code of KVmem and CIPITR to train the model and presented the best result we got. KVmem does not have any beam search, and both CIPITR and our model employ beam search for predicting the action sequences. When inferring the testing samples, we used the top beam [Saha *et al.*, 2019], i.e., the predicted program with the highest probability in the beam to yield the answer.

Table 1 below summarizes the results. We can make a number of important observations.

1. Our full model MARL achieves the best overall performance of 66.96% and 77.71% for macro and micro F1, respectively, outperforming all the baseline models KVmem, CIPITR-All, and CIPITR-Sep. The performance advantage on macro F1 over the three baselines is especially pronounced (47.51, 47.14, and 19.26 percentage points over KVmem, CIPITR-All, and CIPITR-Sep respectively). This is mainly due to the severe imbalance of the CQA dataset. As Table 1 shows, almost 49% (462K/944K) of the CQA training set belongs to the Simple Question category, while four other categories only account for 10% or less each. Given such a large distributional bias, KVmem and CIPITR are unable to learn task-specific knowledge adequately.
2. MARL achieves the best or second-best performance in all the seven categories. Of the three hardest categories (middle part of Table 1), MARL delivers the best performance in all three types. This validates the effectiveness of our meta-learning-based approach in effectively learning task-specific knowledge. Note that the two categories that MARL performs the best, Comparative Reasoning and Comparative (Count), both account for less than 5% of the training set, which further demonstrates MARL's excellent adaptability.
3. CIPITR-Sep achieves the best performance in two *easy* categories, including the largest type, Simple Question. For the three hard categories, it performs poorly compared to other models. This further demonstrates the limitation of coarse-grained adaptation.

4. CIPITR-All, the model that trains over all types of the questions, performs much worse in all the categories than CIPITR-Sep, which learns a different model separately for each question category. For CIPITR-Sep, the results reported for each category are obtained from the models explicitly tuned for that category. A possible reason for CIPITR-All's significant performance degradation is that it is hard for such a one-size-fits-all model to find the weights that fit the training data when the examples vary widely. Besides, the imbalanced classes of questions also deteriorate the performance of the model. Different from CIPITR, our model is designed to adapt appropriately to various categories of questions with one model, thus only needs to be trained once.
5. In general, our MAML-based model variants, Random, Jaccard, and MARL, outperform their non-MAML counterpart, Vanilla, which demonstrates the advantage of our method in learning task-specific knowledge.

We also conducted an ablation study to examine the effectiveness of the retriever. Table 2 presents the result of the ablation study. As can be seen, the random retriever slightly improves performance over the base model Vanilla by 0.97 percentage points, and the fixed retriever Jaccard improves upon Vanilla by 1.59 percentage points. Our full model achieves a 2.99 percentage point improvement over Vanilla. We can observe that the full model improves the overall micro F1 score by 1.40 percentage points compared with the fixed retriever Jaccard. The performance disparity between the full model and the Jaccard retriever can be attributed to our joint training strategy, in which our full model alternately optimizes the programmer and the retriever. On the contrary, less optimally, the Jaccard retriever is separately trained before training the programmer. These results demonstrate the effectiveness of our meta-learning approach as well as the power of our model that jointly optimizes the retriever with the programmer.

4 Related Work

CQA. A behavior cloning based method, Dialog-to-Action (D2A) [Guo *et al.*, 2018], is proposed to answer complex questions by learning from the annotated programs. D2A employs a BFS algorithm to annotate the questions with the corresponding action sequences, and use the annotations to train the programmer. On the other hand, the NPI based methods, i.e., Neural-Symbolic Machines (NSM) [Liang *et al.*, 2017], CIPITR [Saha *et al.*, 2019], and Stable Sparse Reward based Programmer (SSRP)[Ansari *et al.*, 2019], employ the yielded answers as the distant-supervision to learn a programmer. NSM is proposed to answer the multi-hop questions. It first annotates the questions with the pseudo-gold programs and then assigns the annotated programs with a deterministic probability, therefore, to anchor the model to the high-reward programs. CIPITR and SSRP both aim to alleviate the sparse reward problem that appears in conventional NPI approaches and employ high-level constraints to guide the programmer to produce semantically plausible programs. Except for CIPITR, the NPI approaches learn a one-size-fits-all model for the entire dataset. However, CIPITR learns a one-size-fits-all model for each question type instead of empowering the model to learn to transfer from one question type to another. Different from learning a one-size-fits-all model, we aim to learn a model that quickly discovers the knowledge specific to a new task, and employ the acquired knowledge to adapt the programmer to the new task.

Meta-Learning. The Meta-learning approaches utilize the inductive biases, which are meta-learned in learning similar tasks to make the model learn the new task quickly. To make the model sensitive to the new task, one popular direction of meta-learning is to train a meta-learner to learn how to update the parameters of the underlying model [Li and Malik, 2017; Ha *et al.*, 2017], which has been investigated in MAML [Finn *et al.*, 2017]. In semantic parsing tasks, Huang et al [2018] propose a relevance function to find similar samples to form a pseudo-task for each WikiSQL question. Subsequently, they reduce a supervised learning problem into a meta-learning problem and employ MAML to adapt the programmer to each pseudo-task. Likewise, S2A [Guo *et al.*, 2019] separately trains the retriever and the programmer by using the pseudo-gold annotations. Based on the similar samples found by the retriever, S2A establishes a meta-learning task for each question and thus employs MAML to finetune the programmer. Unlike them, we propose a MAML-based approach that trains the retriever and the programmer jointly. It is worth noting that S2A aims to answer conversational questions while we consider answering single-turn questions. Therefore we do not include S2A as a baseline method in the evaluation as it is not directly comparable to our problem setup.

5 Conclusion

In this paper, we presented a novel method for complex question answering over knowledge bases. In a meta-learning framework, our model jointly and alternately optimized a retriever, which learned to select questions, and a programmer, which learned to adapt to the selected secondary questions to produce an answer to a given primary question. Our model was capable of quickly adapting to new questions as it could learn from similar questions. Moreover, it did so from weak supervision signals, the model's performance on question answering. Thus, our model addressed several essential challenges facing existing methods, namely the significant distributional biases present in the dataset and the high cost associated with manual labeling of similar questions. Our evaluation against a number of state-of-the-art models showed the superiority of our model on the large-scale complex question answering dataset CQA. In the future, we plan to extend our model to other domains and tasks that require the manual construction of support sets.

Acknowledgments

Research presented in this paper was partially supported by the National Key Research and Development Program of China under grants (2017YFB1002801, 2018YFC0830200), the Natural Science Foundation of China grants (U1736204, 61602259), Australian Research Council (DP190100006), the Judicial Big Data Research Centre, School of Law at Southeast University, and the project no. 31511120201 and 31510040201.

References

[Ansari *et al.*, 2019] Ghulam Ahmed Ansari, Amrita Saha, Vishwajeet Kumar, Mohan Bhambhani, Karthik Sankaranarayanan, and Soumen Chakrabarti. Neural program induction for kbqa without gold programs or query annotations. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 4890–4896. AAAI Press, 2019.

[Berant *et al.*, 2013] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, 2013.

[Bordes *et al.*, 2015] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*, 2015.

[Finn *et al.*, 2017] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[Guo *et al.*, 2018] Daya Guo, Duyu Tang, Nan Duan, Ming Zhou, and Jian Yin. Dialog-to-action: conversational question answering over a large-scale knowledge base. In *Advances in Neural Information Processing Systems*, pages 2942–2951, 2018.

[Guo *et al.*, 2019] Daya Guo, Duyu Tang, Nan Duan, Ming Zhou, and Jian Yin. Coupling retrieval and meta-learning for context-dependent semantic parsing. In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 855–866. Association for Computational Linguistics, 2019.

[Ha *et al.*, 2017] David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[Huang *et al.*, 2013] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 2333–2338, 2013.

[Huang *et al.*, 2018] Po-Sen Huang, Chenglong Wang, Rishabh Singh, Wen-tau Yih, and Xiaodong He. Natural language to structured query generation via meta-learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 732–738, 2018.

[Li and Malik, 2017] Ke Li and Jitendra Malik. Learning to optimize neural nets. *CoRR*, abs/1703.00441, 2017.

[Liang *et al.*, 2017] Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 23–33, 2017.

[Liu *et al.*, 2019] Shikun Liu, Andrew Davison, and Edward Johns. Self-supervised generalisation with meta auxiliary learning. In *Advances in Neural Information Processing Systems*, pages 1677–1687, 2019.

[Luo *et al.*, 2019] Liangchen Luo, Yuanhao Xiong, Yan Liu, and Xu Sun. Adaptive gradient methods with dynamic bound of learning rate. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[Nichol and Schulman, 2018] Alex Nichol and John Schulman. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2:2, 2018.

[Saha *et al.*, 2018] Amrita Saha, Vardaan Pahuja, Mitesh M Khapra, Karthik Sankaranarayanan, and Sarath Chandar. Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[Saha *et al.*, 2019] Amrita Saha, Ghulam Ahmed Ansari, Abhishek Laddha, Karthik Sankaranarayanan, and Soumen Chakrabarti. Complex program induction for querying knowledge bases in the absence of gold programs. *Transactions of the Association for Computational Linguistics*, 7:185–200, 2019.

[Schmidhuber, 1992] Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992.

[Shen *et al.*, 2019] Tao Shen, Xiubo Geng, Tao QIN, Daya Guo, Duyu Tang, Nan Duan, Guodong Long, and Daxin Jiang. Multi-task learning for conversational question answering over a large-scale knowledge base. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2442–2451, Hong Kong, China, November 2019. Association for Computational Linguistics.

[Yih *et al.*, 2014] Wen-tau Yih, Xiaodong He, and Christopher Meek. Semantic parsing for single-relation question answering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 643–648, 2014.

[Yu *et al.*, 2017] Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. Improved neural relation detection for knowledge base question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 571–581, 2017.

Chapter 6

Conclusion and Future Work

6.1 Conclusions

A KB is a collection of atomic facts about the real world in triples (head, relation, tail) and can be constructed as a big graph. KBs provide an effective data structure for organizing huge amounts of high-quality knowledge and have become an important information source supporting various real-world applications, such as chatbots, search engines, decision support systems, etc. Though formal query languages (such as SPARQL) are considered a standard way to access KB, it is not easy for practical use since they require specialized knowledge of the KB's schema and the language's syntax. To bridge the gap between NLQs and structured KBs, a natural language interface hiding the complexity of KB structure and formal language's grammar—KBQA, has attracted significant attention from academicians and practitioners recently. A KBQA system automatically answers the NLQs posed by users with a direct result, which is obtained by retrieving one or multiple relevant facts from the underlying KB. To express the users' complex information needs, KBQA has evolved from simple questions (the questions that could be answered by using a single fact) to complex questions (the questions that require reasoning over a series of facts to yield the right answers). Therefore, CQA is defined in contrast to simple KBQA, necessitating functional operations on multiple facts across a KB, such as comparison, aggregation, and sorting. Previously proposed approaches to CQA formulate it as an information retrieval task, which produces unexplainable answers, or attempt to employ templates and heuristic rules to express questions' intention and generate answers, which has limited coverage. Also, NSP-based methods formalize the input questions into logical forms with the

help of neural networks, where the supervised learning methods are faced with a high cost for data annotations, and the weak supervised learning paradigm needs to solve the sparse reward problem. Furthermore, the one-size-fits-all neural network models often achieve uneven performance on a wide variety of questions.

This thesis presented our deep learning-based models, which mitigated the above challenges in answering complex questions over large-scale KBs. Specifically, this work is built on a Meta-RL framework and focuses on three sub-problems:

- How to design an NPI-based CQA model that alleviates the sparse reward and data-inefficiency problems?
- How to construct a meta-learning framework to adaptively generates multiple sets of parameters for multiple different tasks?
- How to unify the process of training a retriever and a meta learner into a single deep learning framework when conducting meta-learning?

Our first study, presented in Chapter 3, developed a solution for the first of these problems and proposed a novel framework, **NS-CQA**, for training a CQA model only using the final answer as weak supervision. We designed a set of primitive actions for logical, arithmetic, and aggregation operations required for answering complex questions and proposed an encoder-decoder network to translate the input NLQs into action sequences. We employed a reinforcement learning algorithm to train the network, where the comparison between the execution result of the produced action sequence and the ground-truth answer was used as the weak supervision signals for parameter optimization. Therefore, we married two powerful ideas: continuous representations for language understanding and discrete symbolic program execution for inference. We augmented our RL-based framework with a memory buffer to record promising trials that lead to correct results, addressing the sparse reward and data-inefficiency challenges. By employing a curriculum learning method, we leveraged the trials in the memory buffer to compute the bonus. With the help of the bonus, the model was encouraged to explore the unseen trials in the early stage of the training, and simulated successful trials in the later period. Our empirical studies across two diverse CQA datasets demonstrated that NS-CQA could achieve better performance than other state-of-the-art RL-based CQA approaches.

In Chapter 4, we solved the second problem by extending our RL-based model to a Meta-RL

framework, aka **MRL-CQA**, where a meta-learner was introduced to finetune the CQA model to adapt to varying tasks. We divided the meta-learning process into two steps, i.e., meta-training and meta-testing steps. In the meta-training step, we designed an unsupervised relevance function to construct a support set and finetune the meta-learner, aka the model's initial parameters, to generatethe task-specific parameters to adapt to the support set. In the meta-testing step, we evaluated the task-specific parameters on the meta-testing data to compute a meta-gradient, which involves a gradient through a gradient and are thus used for updating the initial parameters. We performed the gradient descent optimization algorithm used in our RL-based model to update model parameters in both steps. We thus found a set of initial parameters that are sensitive enough where only a few gradient steps on a novel task would lead to a set of task-specific parameters. Empirical studies over a large-scale CQA dataset indicate that our proposed approach is effective as it outperforms state-of-the-art methods significantly. The case study showed that compared with our one-size-fits-all model presented in Chapter 3, **MRL-CQA** could produce more feasible actions and thus alleviate the ambiguity problems that **NS-CQA** was faced with.

Finally, in Chapter 5, we upgraded the unsupervised retriever proposed in Chapter 4 to resolve the last problem, presenting **MARL**. We split the meta-learning process in each training epoch into two stages: training the meta-learner and retriever. In the first stage, we froze the retriever's parameters and performed meta-optimization over the meta-learner, aka the model's initial parameters, as we did in Chapter 4. In the second stage, we fixed the meta-learner and employed an RL approach to optimize the retriever. The retriever sampled M different support sets for the same novel task and generate M sets of task-specific parameters based on the fixed initial parameters. Under the RL paradigm, the retriever was encouraged to produce a high-quality support set, leading to better task-specific parameters and thus obtaining higher rewards. We trained the meta-learner and the retriever alternately until convergence. Therefore, we proposed a retriever that did not rely on the hand-crafted relevance function and could be trained along with the meta-learner. Empirical studies not only showed that **MARL** outperformed other state-of-the-art CQA models (especially our meta-learning based model, **MRL-CQA**) but also shed light on the role that support sets played in meta-learning.

6.2 Future Research Plans

Our deep learning-based CQA models have proven to achieve state-of-the-art performance in several CQA datasets, showing the current research direction has potential for future high-demand information needs. Consequently, we outline a new setting for integrating our models and future directions for addressing new challenges arising from recent developments.

6.2.1 Complex Sequential Question Answering

Our models focus on answering a single-round question, where no context information is considered in answering the current question. In contrast, the Complex Sequential Question Answering (CSQA) over KBs is a different problem setup, where each context-aware question to be answered is part of a multiple-turn conversation. The conversational, natural language question is required to attend to the conversation context when retrieving relevant facts in a KB. Different from answering single-round questions, the ellipsis and co-reference phenomena are frequently encountered in the CSQA task. Therefore, we need to propose a CSQA framework that can: (i) resolve ellipsis or coreference phenomena in conversations by leveraging contextual information, including historical KB artifacts that have been identified in previous utterances, previously generated action sequences, and the obtained intermediate results, (ii) parse the input questions into correct programs, and (iii) reason over the retrieved KB facts to achieve answers. Moreover, in the meta-learning paradigm, a context-aware retriever is needed to construct support sets, where the context environment has been taken into consideration. Our future work will extend our proposed models to adapt to the context-dependent setting and explore the frameworks that are more suited to the CSQA task.

6.2.2 A Retriever with Hierarchical Structure

As depicted in Chapter 5, we employed a filter in the retriever to only consider questions of the same type as a meta-testing example when searching for support sets. With the help of the filter, we partitioned the search space into multiple smaller linear subspaces. However, even we have employed the filtering mechanism, we practically searched the support sets with a linear search that sequentially checks each question of a certain subspace. Therefore, the search space is still too huge. The large

search space will reduce the searching efficiency, making the retriever hard to converge. We will continue to explore new ways of designing the retriever to reduce search space and accelerate searching speed. Compared with the linear data structure, the tree is a hierarchical way to structure data in a multidimensional space. Current studies have already employed tree-based algorithms for the nearest neighbor search. For instance, the ball-tree and the KD tree algorithm have been used for spatial division of data points, allocating the data points into certain regions. For exact nearest-neighbor searches, compared with the linear search, trees can yield great accelerations. Therefore, the ball-tree and KD tree are better alternatives due to their speed and efficiency. Our future work will incorporate the tree-based search algorithms into the framework to design a retriever with a hierarchical structure.

6.2.3 Investigation of Support Set Construction

As described in Chapter 5, we carried out the correlation analysis concerning the influence of separate support set qualities on the performance of meta-learning, demonstrating that the support set with higher quality, i.e., more relevant to the meta-testing data, could improve meta-learning performance. Since our work has discovered the correlation between support sets and meta-learning performance, we can continue to explore the mechanism behind this phenomenon by more closely investigating the effect of support set in meta-learning, attempting to provide some insights into the observations. Meta-learning is highly dependent on episodic training, where a model is trained to adapt to a novel task by learning the specific knowledge from the support set. As a result, we can mine the support sets' characteristic to answer a series of questions, including:

- Will the supporting questions more similar to a query example be more helpful to generate task-specific parameters?
- What kind of supporting questions will allow the meta-gradients to happen in the right direction for fast-learning?
- Will the reduction of support set diversity (for instance reduce the candidate samples used for constructing support sets) always result in adverse effects for meat-learning?

We believe that it is worthwhile to continue investigating the efficacy of support set construction in improving the meta-learning performance, and understand the nature of the phenomenon to pave the way for future work.

Bibliography

[1] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pp. 1247–1250, AcM, 2008.

[2] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann, "DBpedia-a crystallization point for the web of data," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 7, no. 3, pp. 154–165, 2009.

[3] F. Erxleben, M. Günther, M. Krötzsch, J. Mendez, and D. Vrandečić, "Introducing Wikidata to the Linked Data web," in *International Semantic Web Conference*, pp. 50–65, Springer, 2014.

[4] J. Berant, A. Chou, R. Frostig, and P. Liang, "Semantic parsing on freebase from question-answer pairs," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1533–1544, 2013.

[5] X. Yao and B. Van Durme, "Information extraction over structured data: Question answering with freebase," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, pp. 956–966, 2014.

[6] W.-t. Yih, X. He, and C. Meek, "Semantic parsing for single-relation question answering," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, vol. 2, pp. 643–648, 2014.

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

[8] W.-t. Yih, M.-W. Chang, X. He, and J. Gao, "Semantic parsing via staged query graph generation: Question answering with knowledge base," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, vol. 1, pp. 1321–1331, 2015.

[9] M. Yu, W. Yin, K. S. Hasan, C. dos Santos, B. Xiang, and B. Zhou, "Improved neural relation detection for knowledge base question answering," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, pp. 571–581, 2017.

[10] 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 *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, vol. 2, pp. 201–206, 2016.

[11] C. Liang, J. Berant, Q. Le, K. D. Forbus, and N. Lao, "Neural symbolic machines: Learning semantic parsers on freebase with weak supervision," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, pp. 23–33, 2017.

[12] A. Saha, G. A. Ansari, A. Laddha, K. Sankaranarayanan, and S. Chakrabarti, "Complex program induction for querying knowledge bases in the absence of gold programs," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 185–200, 2019.

[13] 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 *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[14] G. A. Ansari, A. Saha, V. Kumar, M. Bhambhani, K. Sankaranarayanan, and S. Chakrabarti, "Neural program induction for kbqa without gold programs or query annotations," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pp. 4890–4896, AAAI Press, 2019.

[15] N. Savinov, A. Raichuk, D. Vincent, R. Marinier, M. Pollefeys, T. P. Lillicrap, and S. Gelly, "Episodic curiosity through reachability," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, OpenReview.net, 2019.

[16] C. Liang, M. Norouzi, J. Berant, Q. V. Le, and N. Lao, "Memory augmented policy optimization for program synthesis and semantic parsing," in *Advances in Neural Information Processing Systems*, pp. 9994–10006, 2018.

[17] P.-S. Huang, C. Wang, R. Singh, W.-t. Yih, and X. He, "Natural language to structured query generation via meta-learning," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 732–738, 2018.

[18] D. Guo, D. Tang, N. Duan, M. Zhou, and J. Yin, "Coupling retrieval and meta-learning for context-dependent semantic parsing," in *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers* (A. Korhonen, D. R. Traum, and L. Màrquez, eds.), pp. 855–866, Association for Computational Linguistics, 2019.

[19] D. Guo, D. Tang, N. Duan, M. Zhou, and J. Yin, "Dialog-to-action: Conversational question answering over a large-scale knowledge base," in *Advances in Neural Information Processing Systems*, pp. 2946–2955, 2018.

[20] A. Neelakantan, Q. V. Le, and I. Sutskever, "Neural programmer: Inducing latent programs with gradient descent," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2016.

[21] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning (ICML'09)*, pp. 41–48, 2009.

[22] M. Fang, T. Zhou, Y. Du, L. Han, and Z. Zhang, "Curriculum-guided hindsight experience replay," in *Advances in Neural Information Processing Systems*, pp. 12602–12613, 2019.

[23] O. Etzioni, "Search needs a shake-up," *Nat.*, vol. 476, no. 7358, pp. 25–26, 2011.

[24] K. Liu, J. Zhao, S. He, and Y. Zhang, "Question answering over knowledge bases," *IEEE Intell. Syst.*, vol. 30, no. 5, pp. 26–35, 2015.

[25] A. Bordes, S. Chopra, and J. Weston, "Question answering with subgraph embeddings," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL* (A. Moschitti, B. Pang, and W. Daelemans, eds.), pp. 615–620, ACL, 2014.

[26] W. Wu, Z. Zhu, G. Zhang, S. Kang, and P. Liu, "A reasoning enhance network for muti-relation question answering," *Applied Intelligence*, pp. 1–10, 2021.

[27] M. Bakhshi, M. Nematbakhsh, M. Mohsenzadeh, and A. M. Rahmani, "Data-driven construction of SPARQL queries by approximate question graph alignment in question answering over knowledge graphs," *Expert Syst. Appl.*, vol. 146, p. 113205, 2020.

[28] A. Kristiadi, M. A. Khan, D. Lukovnikov, J. Lehmann, and A. Fischer, "Incorporating literals into knowledge graph embeddings," in *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part I* (C. Ghidini, O. Hartig, M. Maleshkova, V. Svátek, I. F. Cruz, A. Hogan, J. Song, M. Lefrançois, and F. Gandon, eds.), vol. 11778 of *Lecture Notes in Computer Science*, pp. 347–363, Springer, 2019.

[29] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery, "Learning to construct knowledge bases from the world wide web," *Artificial intelligence*, vol. 118, no. 1-2, pp. 69–113, 2000.

[30] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, *et al.*, "Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia," *Semantic web*, vol. 6, no. 2, pp. 167–195, 2015.

[31] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: a core of semantic knowledge," in *Proceedings of the 16th international conference on World Wide Web*, pp. 697–706, 2007.

[32] G. A. Miller, *WordNet: An electronic lexical database*. MIT press, 1998.

[33] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. H. Jr., and T. M. Mitchell, "Toward an architecture for never-ending language learning," in *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010* (M. Fox and D. Poole, eds.), AAAI Press, 2010.

[34] R. Navigli and S. P. Ponzetto, "Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network," *Artificial Intelligence*, vol. 193, pp. 217–250, 2012.

[35] L. Ji, Y. Wang, B. Shi, D. Zhang, Z. Wang, and J. Yan, "Microsoft concept graph: Mining semantic concepts for short text understanding," *Data Intelligence*, vol. 1, no. 3, pp. 238–270, 2019.

[36] R. Speer, J. Chin, and C. Havasi, "Conceptnet 5.5: An open multilingual graph of general knowledge," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA* (S. P. Singh and S. Markovitch, eds.), pp. 4444–4451, AAAI Press, 2017.

[37] L. P. Heck, D. Hakkani-Tür, and G. Tür, "Leveraging knowledge graphs for web-scale unsupervised semantic parsing," in *INTERSPEECH 2013, 14th Annual Conference of the International Speech Communication Association, Lyon, France, August 25-29, 2013* (F. Bimbot, C. Cerisara, C. Fougeron, G. Gravier, L. Lamel, F. Pellegrino, and P. Perrier, eds.), pp. 1594–1598, ISCA, 2013.

[38] D. Damljanovic and K. Bontcheva, "Named entity disambiguation using linked data," in *Proceedings of the 9th Extended Semantic Web Conference*, pp. 231–240, 2012.

[39] Z. Zheng, X. Si, F. Li, E. Y. Chang, and X. Zhu, "Entity disambiguation with freebase," in *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, vol. 1, pp. 82–89, IEEE, 2012.

[40] R. Hoffmann, C. Zhang, X. Ling, L. Zettlemoyer, and D. S. Weld, "Knowledge-based weak supervision for information extraction of overlapping relations," in *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, pp. 541–550, 2011.

[41] J. Daiber, M. Jakob, C. Hokamp, and P. N. Mendes, "Improving efficiency and accuracy in multilingual entity extraction," in *Proceedings of the 9th International Conference on Semantic Systems*, pp. 121–124, 2013.

[42] A. Bordes, J. Weston, and N. Usunier, "Open question answering with weakly supervised embedding models," in *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part I* (T. Calders, F. Esposito, E. Hüllermeier, and R. Meo, eds.), vol. 8724 of *Lecture Notes in Computer Science*, pp. 165–180, Springer, 2014.

[43] L. Ehrlinger and W. Wöß, "Towards a definition of knowledge graphs.," *SEMANTiCS (Posters, Demos, SuCCESS)*, vol. 48, no. 1-4, p. 2, 2016.

[44] H. Wang, Z. Fang, and T. Ruan, "Kcf. js: a javascript library for knowledge cards fusion," in *Proceedings of the 25th International Conference Companion on World Wide Web*, pp. 267–270, 2016.

[45] R. Das, M. Zaheer, D. Thai, A. Godbole, E. Perez, J.-Y. Lee, L. Tan, L. Polymenakos, and A. McCallum, "Case-based reasoning for natural language queries over knowledge bases," *arXiv preprint arXiv:2104.08762*, 2021.

[46] L. Hirschman and R. Gaizauskas, "Natural language question answering: the view from here," *natural language engineering*, vol. 7, no. 4, pp. 275–300, 2001.

[47] S. Vodanovich, D. Sundaram, and M. Myers, "Research commentary—digital natives and ubiquitous information systems," *Information Systems Research*, vol. 21, no. 4, pp. 711–723, 2010.

[48] D. Radev, W. Fan, H. Qi, H. Wu, and A. Grewal, "Probabilistic question answering on the web," in *Proceedings of the 11th international conference on World Wide Web*, pp. 408–419, 2002.

[49] D. Roussinov and J. A. Robles-Flores, "Applying question answering technology to locating malevolent online content," *Decision Support Systems*, vol. 43, no. 4, pp. 1404–1418, 2007.

[50] D. A. Ferrucci, "Introduction to "this is watson"," *IBM Journal of Research and Development*, vol. 56, no. 3.4, pp. 1–1, 2012.

[51] B. Kratzwald and S. Feuerriegel, "Putting question-answering systems into practice: Transfer learning for efficient domain customization," *ACM Transactions on Management Information Systems (TMIS)*, vol. 9, no. 4, pp. 1–20, 2019.

[52] Y. Cao, F. Liu, P. Simpson, L. Antieau, A. Bennett, J. J. Cimino, J. Ely, and H. Yu, "Askhermes: An online question answering system for complex clinical questions," *Journal of biomedical informatics*, vol. 44, no. 2, pp. 277–288, 2011.

[53] J. Cao and J. F. Nunamaker, "Question answering on lecture videos: a multifaceted approach," in *Proceedings of the 2004 Joint ACM/IEEE Conference on Digital Libraries, 2004.*, pp. 214–215, IEEE, 2004.

[54] F.-L. Li, W. Chen, Q. Huang, and Y. Guo, "Alime kbqa: Question answering over structured knowledge for e-commerce customer service," in *China Conference on Knowledge Graph and Semantic Computing*, pp. 136–148, Springer, 2019.

[55] B. Fu, Y. Qiu, C. Tang, Y. Li, H. Yu, and J. Sun, "A survey on complex question answering over knowledge base: Recent advances and challenges," *arXiv preprint arXiv:2007.13069*, 2020.

[56] R. J. Kate and R. J. Mooney, "Using string-kernels for learning semantic parsers," in *ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July 2006* (N. Calzolari, C. Cardie, and P. Isabelle, eds.), The Association for Computer Linguistics, 2006.

[57] J. Cheng, S. Reddy, V. Saraswat, and M. Lapata, "Learning an executable neural semantic parser," *Computational Linguistics*, vol. 45, no. 1, pp. 59–94, 2019.

[58] A. Saha, G. A. Ansari, A. Laddha, K. Sankaranarayanan, and S. Chakrabarti, "Complex program induction for querying knowledge bases in the absence of gold programs," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 185–200, 2019.

[59] Y. Zhang, K. Liu, S. He, G. Ji, Z. Liu, H. Wu, and J. Zhao, "Question answering over knowledge base with neural attention combining global knowledge information," *CoRR*, vol. abs/1606.00979, 2016.

[60] X. Yin, D. Gromann, and S. Rudolph, "Neural machine translating from natural language to sparql," *Future Generation Computer Systems*, vol. 117, pp. 510–519, 2019.

[61] P. Trivedi, G. Maheshwari, M. Dubey, and J. Lehmann, "Lc-quad: A corpus for complex question answering over knowledge graphs," in *International Semantic Web Conference*, pp. 210–218, Springer, 2017.

[62] C. Unger, L. Bühmann, J. Lehmann, A.-C. Ngonga Ngomo, D. Gerber, and P. Cimiano, "Template-based question answering over rdf data," in *Proceedings of the 21st international conference on World Wide Web*, pp. 639–648, 2012.

[63] S. Shekarpour, A.-C. Ngonga Ngomo, and S. Auer, "Question answering on interlinked data," in *Proceedings of the 22nd international conference on World Wide Web*, pp. 1145–1156, 2013.

[64] M. Yahya, K. Berberich, S. Elbassuoni, and G. Weikum, "Robust question answering over the web of linked data," in *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pp. 1107–1116, 2013.

[65] J. Bao, N. Duan, M. Zhou, and T. Zhao, "Knowledge-based question answering as machine translation," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, pp. 967–976, 2014.

[66] P. Liang, M. I. Jordan, and D. Klein, "Learning dependency-based compositional semantics," in *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA* (D. Lin, Y. Matsumoto, and R. Mihalcea, eds.), pp. 590–599, The Association for Computer Linguistics, 2011.

[67] J. Johnson, B. Hariharan, L. van der Maaten, J. Hoffman, L. Fei-Fei, C. L. Zitnick, and R. B. Girshick, "Inferring and executing programs for visual reasoning," in *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pp. 3008–3017, IEEE Computer Society, 2017.

[68] N. Chakraborty, D. Lukovnikov, G. Maheshwari, P. Trivedi, J. Lehmann, and A. Fischer, "Introduction to neural network based approaches for question answering over knowledge graphs," *CoRR*, vol. abs/1907.09361, 2019.

[69] T. Shen, X. Geng, T. Qin, D. Guo, D. Tang, N. Duan, G. Long, and D. Jiang, "Multi-task learning for conversational question answering over a large-scale knowledge base," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th*

International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019 (K. Inui, J. Jiang, V. Ng, and X. Wan, eds.), pp. 2442–2451, Association for Computational Linguistics, 2019.

[70] T. Shen, X. Geng, G. Long, J. Jiang, C. Zhang, and D. Jiang, "Effective search of logical forms for weakly supervised knowledge-based question answering," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020* (C. Bessiere, ed.), pp. 2227–2233, ijcai.org, 2020.

[71] O. Goldman, V. Latcinnik, E. Nave, A. Globerson, and J. Berant, "Weakly supervised semantic parsing with abstract examples," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1809–1819, 2018.

[72] A. Kamath and R. Das, "A survey on semantic parsing," in *1st Conference on Automated Knowledge Base Construction, AKBC 2019, Amherst, MA, USA, May 20-22, 2019*, 2019.

[73] P. Pasupat and P. Liang, "Inferring logical forms from denotations," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*, The Association for Computer Linguistics, 2016.

[74] K. Guu, P. Pasupat, E. Z. Liu, and P. Liang, "From language to programs: Bridging reinforcement learning and maximum marginal likelihood," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers* (R. Barzilay and M. Kan, eds.), pp. 1051–1062, Association for Computational Linguistics, 2017.

[75] S. Vakulenko, J. D. Fernandez Garcia, A. Polleres, M. de Rijke, and M. Cochez, "Message passing for complex question answering over knowledge graphs," in *Proceedings of the 28th acm international conference on information and knowledge management*, pp. 1431–1440, 2019.

[76] Ó. Ferrández, C. Spurk, M. Kouylekov, I. Dornescu, S. Ferrández, M. Negri, R. Izquierdo, D. Tomás, C. Orasan, G. Neumann, *et al.*, "The qall-me framework: A specifiable-domain multilingual question answering architecture," *Journal of web semantics*, vol. 9, no. 2, pp. 137–145, 2011.

[77] J.-D. Kim, C. Unger, A.-C. N. Ngomo, A. Freitas, Y. G. Hahm, J. Kim, G.-H. Choi, J.-U. Kim, R. Usbeck, M.-G. Kang, *et al.*, "Okbqa: an open collaboration framework for development of natural language question-answering over knowledge bases," in *2017 ISWC Posters and Demonstrations and Industry Tracks, ISWC-P and D-Industry 2017*, Semantic Web Science Association, 2017.

[78] K. Singh, A. Both, A. Sethupat, and S. Shekarpour, "Frankenstein: A platform enabling reuse of question answering components," in *European Semantic Web Conference*, pp. 624–638, Springer, 2018.

[79] H. Isozaki and H. Kazawa, "Efficient support vector classifiers for named entity recognition," in *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002.

[80] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF models for sequence tagging," *arXiv preprint arXiv:1508.01991*, 2015.

[81] X. Huang, J. Zhang, D. Li, and P. Li, "Knowledge graph embedding based question answering," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pp. 105–113, 2019.

[82] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer, "Dbpedia spotlight: shedding light on the web of documents," in *Proceedings of the 7th international conference on semantic systems*, pp. 1–8, 2011.

[83] Y. Yang and M.-W. Chang, "S-mart: Novel tree-based structured learning algorithms applied to tweet entity linking," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 504–513, 2015.

[84] C. Zhu, K. Ren, X. Liu, H. Wang, Y. Tian, and Y. Yu, "A graph traversal based approach to answer non-aggregation questions over dbpedia," in *Semantic Technology - 5th Joint International Conference, JIST 2015, Yichang, China, November 11-13, 2015, Revised Selected Papers* (G. Qi, K. Kozaki, J. Z. Pan, and S. Yu, eds.), vol. 9544 of *Lecture Notes in Computer Science*, pp. 219–234, Springer, 2015.

[85] D. Milne and I. H. Witten, "An open-source toolkit for mining wikipedia," *Artificial Intelligence*, vol. 194, pp. 222–239, 2013.

[86] M. Dubey, D. Banerjee, D. Chaudhuri, and J. Lehmann, "Earl: joint entity and relation linking for question answering over knowledge graphs," in *International Semantic Web Conference*, pp. 108–126, Springer, 2018.

[87] Y. Yuan, X. Zhou, S. Pan, Q. Zhu, Z. Song, and L. Guo, "A relation-specific attention network for joint entity and relation extraction," in *International Joint Conference on Artificial Intelligence 2020*, pp. 4054–4060, Association for the Advancement of Artificial Intelligence (AAAI), 2020.

[88] P. Liang, M. I. Jordan, and D. Klein, "Learning dependency-based compositional semantics," *Comput. Linguistics*, vol. 39, no. 2, pp. 389–446, 2013.

[89] J. M. Zelle and R. J. Mooney, "Learning to parse database queries using inductive logic programming," in *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 96, IAAI 96, Portland, Oregon, USA, August 4-8, 1996, Volume 2* (W. J. Clancey and D. S. Weld, eds.), pp. 1050–1055, AAAI Press / The MIT Press, 1996.

[90] I. Androutsopoulos, G. Ritchie, and P. Thanisch, "Masque/sql: An efficient and portable natural language query interface for relational databases," *Database technical paper, Department of AI, University of Edinburgh*, 1993.

[91] F. Holzschuher and R. Peinl, "Performance of graph query languages: comparison of cypher, gremlin and native access in neo4j," in *Joint 2013 EDBT/ICDT Conferences, EDBT/ICDT '13, Genoa, Italy, March 22, 2013, Workshop Proceedings* (G. Guerrini, ed.), pp. 195–204, ACM, 2013.

[92] R. J. Kate, Y. W. Wong, and R. J. Mooney, "Learning to transform natural to formal languages," in *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA* (M. M. Veloso and S. Kambhampati, eds.), pp. 1062–1068, AAAI Press / The MIT Press, 2005.

[93] J. Pérez, M. Arenas, and C. Gutiérrez, "Semantics and complexity of SPARQL," *ACM Trans. Database Syst.*, vol. 34, no. 3, pp. 16:1–16:45, 2009.

[94] H. P. Barendregt and E. Barendsen, "Introduction to lambda calculus," *Nieuw Archief voor Wiskunde*, vol. 4, no. 2, pp. 337–372, 1984.

[95] P. Liang, "Lambda dependency-based compositional semantics," *CoRR*, vol. abs/1309.4408, 2013.

[96] C. Bizer, T. Heath, and T. Berners-Lee, "Linked data - the story so far," *Int. J. Semantic Web Inf. Syst.*, vol. 5, no. 3, pp. 1–22, 2009.

[97] O. Hartig, "An introduction to SPARQL and queries over linked data," in *Web Engineering - 12th International Conference, ICWE 2012, Berlin, Germany, July 23-27, 2012. Proceedings* (M. Brambilla, T. Tokuda, and R. Tolksdorf, eds.), vol. 7387 of *Lecture Notes in Computer Science*, pp. 506–507, Springer, 2012.

[98] S. Ferré, "SQUALL: the expressiveness of SPARQL 1.1 made available as a controlled natural language," *Data Knowl. Eng.*, vol. 94, pp. 163–188, 2014.

[99] A. Church, "An unsolvable problem of elementary number theory," *American journal of mathematics*, vol. 58, no. 2, pp. 345–363, 1936.

[100] Z. Csörnyei and G. Dévai, "An introduction to the lambda calculus," in *Central European Functional Programming School, Second Summer School, CEFP 2007, Cluj-Napoca, Romania, June 23-30, 2007, Revised Selected Lectures* (Z. Horváth, R. Plasmeijer, A. Soós, and V. Zsók, eds.), vol. 5161 of *Lecture Notes in Computer Science*, pp. 87–111, Springer, 2007.

[101] L. Dong and M. Lapata, "Language to logical form with neural attention," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 33–43, 2016.

[102] R. Rojas, "A tutorial introduction to the lambda calculus," *CoRR*, vol. abs/1503.09060, 2015.

[103] T. Kwiatkowski, L. S. Zettlemoyer, S. Goldwater, and M. Steedman, "Lexical generalization in CCG grammar induction for semantic parsing," in *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*, pp. 1512–1523, ACL, 2011.

[104] R. J. Brachman and H. J. Levesque, *Knowledge Representation and Reasoning*. Elsevier, 2004.

[105] M. Benedikt, "An insider's guide to logic in telecommunications data," in *20th IEEE Symposium on Logic in Computer Science (LICS 2005), 26-29 June 2005, Chicago, IL, USA, Proceedings*, pp. 104–105, IEEE Computer Society, 2005.

[106] E. Parisotto, A. Mohamed, R. Singh, L. Li, D. Zhou, and P. Kohli, "Neuro-symbolic program synthesis," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017.

[107] A. Rothe, B. M. Lake, and T. M. Gureckis, "Question asking as program generation," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA* (I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, eds.), pp. 1046–1055, 2017.

[108] J. R. Anderson, R. Farrell, and R. Sauers, "Learning to program in lisp," *Cognitive Science*, vol. 8, no. 2, pp. 87–129, 1984.

[109] O. Goldman, V. Latcinnik, E. Nave, A. Globerson, and J. Berant, "Weakly supervised semantic parsing with abstract examples," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers* (I. Gurevych and Y. Miyao, eds.), pp. 1809–1819, Association for Computational Linguistics, 2018.

[110] A. Suhr, M. Lewis, J. Yeh, and Y. Artzi, "A corpus of natural language for visual reasoning," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 2: Short Papers* (R. Barzilay and M. Kan, eds.), pp. 217–223, Association for Computational Linguistics, 2017.

[111] K. Yi, J. Wu, C. Gan, A. Torralba, P. Kohli, and J. Tenenbaum, "Neural-symbolic VQA: disentangling reasoning from vision and language understanding," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada* (S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), pp. 1039–1050, 2018.

[112] P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer, "The mathematics of statistical machine translation: Parameter estimation," *Computational linguistics*, vol. 19, no. 2, pp. 263–311, 1993.

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

[114] 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 *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 221–231, 2017.

[115] A. Saxena, A. Tripathi, and P. Talukdar, "Improving multi-hop question answering over knowledge graphs using knowledge base embeddings," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4498–4507, 2020.

[116] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *International Conference on Machine Learning*, pp. 2071–2080, PMLR, 2016.

[117] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[118] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus, "Weakly supervised memory networks," *CoRR*, vol. abs/1503.08895, 2015.

[119] J. Weston, S. Chopra, and A. Bordes, "Memory networks," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015.

[120] A. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston, "Key-value memory networks for directly reading documents," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1400–1409, 2016.

[121] Y. Chen, L. Wu, and M. J. Zaki, "Bidirectional attentive memory networks for question answering over knowledge bases," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2913–2923, 2019.

[122] H. Bast and E. Haussmann, "More accurate question answering on freebase," in *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015* (J. Bailey, A. Moffat, C. C. Aggarwal, M. de Rijke, R. Kumar, V. Murdock, T. K. Sellis, and J. X. Yu, eds.), pp. 1431–1440, ACM, 2015.

[123] 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, 2018.

[124] A. Abujabal, M. Yahya, M. Riedewald, and G. Weikum, "Automated template generation for question answering over knowledge graphs," in *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017* (R. Barrett, R. Cummings, E. Agichtein, and E. Gabrilovich, eds.), pp. 1191–1200, ACM, 2017.

[125] W. Zheng, J. X. Yu, L. Zou, and H. Cheng, "Question answering over knowledge graphs: Question understanding via template decomposition," *Proc. VLDB Endow.*, vol. 11, no. 11, pp. 1373–1386, 2018.

[126] W. Cui, Y. Xiao, H. Wang, Y. Song, S. Hwang, and W. Wang, "KBQA: learning question answering over QA corpora and knowledge bases," *Proc. VLDB Endow.*, vol. 10, no. 5, pp. 565–576, 2017.

[127] S. Reddy, M. Lapata, and M. Steedman, "Large-scale semantic parsing without question-answer pairs," *Transactions of the Association for Computational Linguistics*, vol. 2, pp. 377–392, 2014.

[128] S. Clark and J. R. Curran, "Wide-coverage efficient statistical parsing with ccg and log-linear models," *Computational Linguistics*, vol. 33, no. 4, pp. 493–552, 2007.

[129] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a" siamese" time delay neural network," *Advances in neural information processing systems*, pp. 737–737, 1994.

[130] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil, "A latent semantic model with convolutional-pooling structure for information retrieval," in *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*, pp. 101–110, 2014.

[131] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil, "Learning semantic representations using convolutional neural networks for web search," in *Proceedings of the 23rd international conference on world wide web*, pp. 373–374, 2014.

[132] C. J. Burges, "From ranknet to lambdarank to lambdamart: An overview," *Learning*, vol. 11, no. 23-581, p. 81, 2010.

[133] J. Bao, N. Duan, Z. Yan, M. Zhou, and T. Zhao, "Constraint-based question answering with knowledge graph," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pp. 2503–2514, 2016.

[134] K. Luo, F. Lin, X. Luo, and K. Zhu, "Knowledge base question answering via encoding of complex query graphs," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2185–2194, 2018.

[135] G. Maheshwari, P. Trivedi, D. Lukovnikov, N. Chakraborty, A. Fischer, and J. Lehmann, "Learning to rank query graphs for complex question answering over knowledge graphs," in *International semantic web conference*, pp. 487–504, Springer, 2019.

[136] S. Hu, L. Zou, and X. Zhang, "A state-transition framework to answer complex questions over knowledge base," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2098–2108, 2018.

[137] N. Bhutani, X. Zheng, and H. Jagadish, "Learning to answer complex questions over knowledge bases with query composition," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 739–748, 2019.

[138] A. Talmor and J. Berant, "The web as a knowledge-base for answering complex questions," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 641–651, 2018.

[139] 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," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 918–924, 2018.

[140] Y. Chen, H. Li, Y. Hua, and G. Qi, "Formal query building with query structure prediction for complex question answering over knowledge base.," in *IJCAI*, pp. 3751–3758, 2020.

[141] R. Koncel-Kedziorski, D. Bekal, Y. Luan, M. Lapata, and H. Hajishirzi, "Text generation from knowledge graphs with graph transformers," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2284–2293, 2019.

[142] 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 in neural information processing systems*, pp. 5998–6008, 2017.

[143] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, *et al.*, "Huggingface's transformers: State-of-the-art natural language processing," *arXiv preprint arXiv:1910.03771*, 2019.

[144] T. Wolf, J. Chaumond, L. Debut, V. Sanh, C. Delangue, A. Moi, P. Cistac, M. Funtowicz, J. Davison, S. Shleifer, *et al.*, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, 2020.

[145] C. E. Rasmussen and Z. Ghahramani, "Occam's razor," *Advances in neural information processing systems*, pp. 294–300, 2001.

[146] Y. Lan and J. Jiang, "Query graph generation for answering multi-hop complex questions from knowledge bases," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020* (D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, eds.), pp. 969–974, Association for Computational Linguistics, 2020.

[147] Y. Zhang, H. Dai, Z. Kozareva, A. J. Smola, and L. Song, "Variational reasoning for question answering with knowledge graph," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018* (S. A. McIlraith and K. Q. Weinberger, eds.), pp. 6069–6076, AAAI Press, 2018.

[148] R. Das, S. Dhuliawala, M. Zaheer, L. Vilnis, I. Durugkar, A. Krishnamurthy, A. Smola, and A. McCallum, "Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning," in *International Conference on Learning Representations*, 2018.

[149] X. V. Lin, R. Socher, and C. Xiong, "Multi-hop knowledge graph reasoning with reward shaping," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018* (E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, eds.), pp. 3243–3253, Association for Computational Linguistics, 2018.

[150] Y. Qiu, Y. Wang, X. Jin, and K. Zhang, "Stepwise reasoning for multi-relation question answering over knowledge graph with weak supervision," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, pp. 474–482, 2020.

[151] Y. Qiu, K. Zhang, Y. Wang, X. Jin, L. Bai, S. Guan, and X. Cheng, "Hierarchical query graph generation for complex question answering over knowledge graph," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 1285–1294, 2020.

[152] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[153] Y. Wang and H. Jin, "A deep reinforcement learning based multi-step coarse to fine question answering (mscqa) system," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 7224–7232, 2019.

[154] R. Paulus, C. Xiong, and R. Socher, "A deep reinforced model for abstractive summarization," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, OpenReview.net, 2018.

[155] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel, "Self-critical sequence training for image captioning," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 1179–1195, IEEE Computer Society, 2017.

[156] L. Song, Z. Wang, and W. Hamza, "A unified query-based generative model for question generation and question answering," *CoRR*, vol. abs/1709.01058, 2017.

[157] K. Li and J. Malik, "Learning to optimize neural nets," *CoRR*, vol. abs/1703.00441, 2017.

[158] D. Ha, A. M. Dai, and Q. V. Le, "Hypernetworks," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017.

[159] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *CoRR*, vol. abs/1803.02999, 2018.

[160] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, OpenReview.net, 2018.

[161] T. Munkhdalai and H. Yu, "Meta networks," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, pp. 2554–2563, PMLR, 2017.

[162] J. Baxter, "A model of inductive bias learning," *J. Artif. Intell. Res.*, vol. 12, pp. 149–198, 2000.

[163] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, "Human-level concept learning through probabilistic program induction," *Science*, vol. 350, no. 6266, pp. 1332–1338, 2015.

[164] H. Ye, X. Sheng, and D. Zhan, "Few-shot learning with adaptively initialized task optimizer: a practical meta-learning approach," *Mach. Learn.*, vol. 109, no. 3, pp. 643–664, 2020.

[165] L. A. Schmidt, *Meaning and compositionality as statistical induction of categories and constraints*. PhD thesis, Massachusetts Institute of Technology, 2009.

[166] R. Salakhutdinov, J. Tenenbaum, and A. Torralba, "One-shot learning with a hierarchical nonparametric bayesian model," in *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, pp. 195–206, JMLR Workshop and Conference Proceedings, 2012.

[167] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, "Dueling network architectures for deep reinforcement learning," in *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016* (M. Balcan and K. Q. Weinberger, eds.), vol. 48 of *JMLR Workshop and Conference Proceedings*, pp. 1995–2003, JMLR.org, 2016.

[168] H. F. Harlow, "The formation of learning sets.," *Psychological review*, vol. 56, no. 1, p. 51, 1949.

[169] R. Vilalta and Y. Drissi, "A perspective view and survey of meta-learning," *Artificial intelligence review*, vol. 18, no. 2, pp. 77–95, 2002.

[170] A. Maurer, M. Pontil, and B. Romera-Paredes, "The benefit of multitask representation learning," *Journal of Machine Learning Research*, vol. 17, no. 81, pp. 1–32, 2016.

[171] W. Chao, H. Ye, D. Zhan, M. E. Campbell, and K. Q. Weinberger, "Revisiting meta-learning as supervised learning," *CoRR*, vol. abs/2002.00573, 2020.

[172] T. M. Mitchell and S. Thrun, "Explanation-based neural network learning for robot control," in *Advances in Neural Information Processing Systems 5, [NIPS Conference, Denver, Colorado, USA, November 30 - December 3, 1992]* (S. J. Hanson, J. D. Cowan, and C. L. Giles, eds.), pp. 287–294, Morgan Kaufmann, 1992.

[173] R. Vilalta and Y. Drissi, "A perspective view and survey of meta-learning," *Artif. Intell. Rev.*, vol. 18, no. 2, pp. 77–95, 2002.

[174] J. Schmidhuber, "Evolutionary principles in self-referential learning, or on learning how to learn," *Ph. D. dissertation*, 1987.

[175] S. Hochreiter, A. S. Younger, and P. R. Conwell, "Learning to learn using gradient descent," in *International Conference on Artificial Neural Networks*, pp. 87–94, Springer, 2001.

[176] M. Andrychowicz, M. Denil, S. G. Colmenarejo, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas, "Learning to learn by gradient descent by gradient descent," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain* (D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, eds.), pp. 3981–3989, 2016.

[177] K. Li and J. Malik, "Learning to optimize," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017.

[178] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017.

[179] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, pp. 1126–1135, PMLR, 2017.

[180] A. Nichol and J. Schulman, "Reptile: a scalable metalearning algorithm," *arXiv preprint arXiv:1803.02999*, vol. 2, no. 2, p. 1, 2018.

[181] J. Gu, Y. Wang, Y. Chen, V. O. K. Li, and K. Cho, "Meta-learning for low-resource neural machine translation," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018* (E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, eds.), pp. 3622–3631, Association for Computational Linguistics, 2018.

[182] A. Madotto, Z. Lin, C.-S. Wu, and P. Fung, "Personalizing dialogue agents via meta-learning," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5454–5459, 2019.

[183] Y. Sun, D. Tang, N. Duan, Y. Gong, X. Feng, B. Qin, and D. Jiang, "Neural semantic parsing in low-resource settings with back-translation and meta-learning," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 8960–8967, AAAI Press, 2020.

[184] V. Zhong, C. Xiong, and R. Socher, "Seq2sql: Generating structured queries from natural language using reinforcement learning," *CoRR*, vol. abs/1709.00103, 2017.

[185] P. Pasupat and P. Liang, "Compositional semantic parsing on semi-structured tables," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, vol. 1, pp. 1470–1480, 2015.

[186] J. Clarke, D. Goldwasser, M.-W. Chang, and D. Roth, "Driving semantic parsing from the world's response," in *Proceedings of the fourteenth conference on computational natural language learning*, pp. 18–27, 2010.

[187] W. Xiong, X. L. Li, S. Iyer, J. Du, P. S. H. Lewis, W. Y. Wang, Y. Mehdad, W. Yih, S. Riedel, D. Kiela, and B. Oguz, "Answering complex open-domain questions with multi-hop dense retrieval," *CoRR*, vol. abs/2009.12756, 2020.

[188] S. Bengio, Y. Bengio, J. Cloutier, and J. Gecsei, "On the optimization of a synaptic learning rule," in *Optimality in Artificial and Biological Neural Networks*, vol. 2, 1992.

[189] J. Schmidhuber, "Learning to control fast-weight memories: An alternative to dynamic recurrent networks," *Neural Computation*, vol. 4, no. 1, pp. 131–139, 1992.

[190] B. M. Lake, R. Salakhutdinov, J. Gross, and J. B. Tenenbaum, "One shot learning of simple visual concepts," in *Proceedings of the 33th Annual Meeting of the Cognitive Science Society, CogSci 2011, Boston, Massachusetts, USA, July 20-23, 2011* (L. A. Carlson, C. Hölscher, and T. F. Shipley, eds.), cognitivesciencesociety.org, 2011.

[191] A. Setlur, O. Li, and V. Smith, "Is support set diversity necessary for meta-learning?," *CoRR*, vol. abs/2011.14048, 2020.