University of Zurich[UZH]

Natural Language Question Answering via Knowledge Graph Reasoning

Thesis December 21, 2021

Fan Feng
of Zurich ZH, Switzerland

Student-ID: 18-745-414
fan.feng@uzh.ch

Advisor: **Ruijie Wang**

Prof. Abraham Bernstein, PhD
Institut für Informatik
Universität Zürich
http://www.ifi.uzh.ch/ddis

Acknowledgements

I would like to thank Prof. Abraham Bernstein for the precious opportunity to write this master thesis on knowledge graph question answering. I am very much grateful to my supervisor Ruijie Wang for being supportive and responsive all the time. It has been a journey full of pleasure and gains, and I would appreciate for the numerous insightful suggestions and warm words of encouragements. I also want to say a thank you to my friend, Joyce, thanks for the companionship, the boxing classes, the snowy days, and the memories that never fade off the time. Lastly, I will thank my family especially, they are always the most solid backing of mine.

Zusammenfassung

Knowledge Graphs (KGs) haben in den letzten Jahren große Aufmerksamkeit in der Forschung erregt, da sie es ermöglichen, halbstrukturierte Informationen auf einheitliche, verbundene und organisierte Weise zu speichern. Die inhärenten Merkmale dieser Datenstruktur werden bei vielen Aufgaben genutzt, z. B. bei der Informationsbeschaffung, bei Empfehlungssystemen usw.

Gleichzeitig gibt es Herausforderungen beim Verstehen und Schlussfolgern auf einer Teilmenge einer KG. Ein Szenario wäre die Beantwortung von Fragen über KGs. Fragen in natürlicher Sprache können in ihren Ausdrücken flexibel sein, was bedeutet, dass es für Maschinen schwierig ist, eine Antwort aus einem KG zu finden, wenn eine Frage von einem Menschen gestellt wurde.

[Qiu et al., 2020] schlug einen auf Reinforcement Learning (RL) basierenden Ansatz vor, der Antwortentitäten für Multi-Hop-Fragen durch schrittweises Schlussfolgern über KGs findet. Inspiriert von dieser Arbeit, übernimmt diese Arbeit den Hauptteil des Modells als Basisarchitektur und untersucht drei Forschungsfragen.

Die Prämisse des KG-Reasonings ist die genaue Auswahl von Themenentitäten. In dieser Arbeit wird ein passiver Entity Linker angepasst, um Frageerwähnungen mit KG-Knoten zu verknüpfen. In Schlussfolgerungsprozessen wird ein Aufmerksamkeitsmechanismus implementiert, um die Historie von Aktionen mit semantischen Informationen aus Fragen zu verknüpfen, so dass ein Agent lernen kann, auf welchen Teil von Fragen er sich konzentrieren muss.

Herkömmliche RL-basierte Argumentation gibt nach abgeschlossenen Argumentationsepisoden eine endgültige Belohnung zurück, was zu einem Mangel an Orientierung im sequenziellen Entscheidungsprozess führt. Um dieses Problem zu beheben, verwenden wir stattdessen potenzialbasierte Shaping-Belohnungen. Die empirischen Ergebnisse zeigen, dass die Reward-Shaping-Funktion die Leistung von hits@1 bei zwei Benchmarks verbessert.

Abstract

Knowledge graphs (KGs) have drawn a wide research attention in recent years, since they enable semi-structured information to be stored in an unified, connected and organized way. The inherent features of this data structure are leveraged in many tasks, such as information retrieval, recommendation systems, etc.

Meanwhile, there are challenges in understanding and reasoning on a subset of a KG. One scenario would be question answering over KGs. Natural language questions can be flexible in expressions, which means that it is difficult for machines to retrieve an answer from a KG given a question posed by human.

[Qiu et al., 2020] proposed a reinforcement learning-based (RL-based) approach, which finds answer entities for multi-hop questions via stepwise reasoning over KGs. Inspired by its work, this thesis adopts the model's main body as a baseline architecture and investigates three research questions.

The premise of KG reasoning is the accurate selection of topic entities. This work adapts a passive entity linker to link question mentions to KG nodes. In reasoning processes, an attention mechanism is implemented to associate history of actions with semantic information from questions, such that an agent can learn on which part of questions to focus.

Conventional RL-based reasoning returns terminal rewards after complete reasoning episodes, resulting in a lack of guidance in sequential decision process. To address this problem, we use potential-based shaping rewards instead. The empirical results show that the reward shaping function improves the hits@1 performances on two benchmarks.

Contents

1

Introduction

Knowledge graphs (KGs) serve as an unique and vital form of data structure in knowledge representation. KGs extract human knowledge into graph-structured datasets, which are typically composed of fact triples consisting of entities and relations, e.g., (Bern, capital of, Switzerland). In fact triples, entities can be real-world objects or abstract concepts, and relations denote the relationships among entities.

Large-scale KGs have been developed to organize massive real-world information in a consistent, structured, and connected manner. There are many widely known applications, including open KGs, such as DBPedia[1], Freebase[2], and YAGO[3], as well as proprietary KGs, such as Google's knowledge graph[4], which in fact has introduced the term "Knowledge Graph" for the first time in 2012 [Fensel et al., 2020].

Over the past few years, emerging advanced techniques on KGs, e.g., KG embedding models and KG reasoning, have promoted the development of various applications in real world. The rich semantic information stored in KGs motivates people from both academia and industry to explore how to utilize this type of structured knowledge and how to further improve their own products or services. The tasks of KG applications fall into several categories, e.g., recommendation systems, question answering, information retrieval, etc.

Among those research areas, question answering on knowledge graphs (KGQA) refers to a task, that answers are automatically extracted from KGs given natural language questions [Fu et al., 2020]. For example, the question "*who directed the film the Truman Show*?" expects for an entity name "Peter Weir". In general, this kind of single-hop questions can be resolved by matching with KG triples in one go. However, dealing with other complex questions should be a way more troublesome work, because multi-hop relations are involved in it, such as the query "*what were the release dates of films starred by actors in Rain Man?*".

In previous studies, efforts have been made on semantic parsing-based (SP-based) methods and information retrieval-based (IR-based) methods. SP-based approaches heavily rely on either manually defined templates and rules or neural networks, in order

[1] *http://dbpedia.org/*
[2] *http://www.freebase.com/*
[3] *https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/downloads/*
[4] *https://developers.google.com/knowledge-graph/*

to parse natural language questions and to get logical forms. IR-based approaches constrain query scopes to subgraphs surrounding topic entities, and leverage features from questions as well as context information in the subgraph. One representative type of IR-based methods is based upon representation learning mechanism, that maps questions and answer entities to vectors and calculates semantic similarities between them [Fu et al., 2020].

Since SP-based solutions require sophisticated query parsing and high-quality logic forms, which might be difficult to master for non-expert users, some researchers are particularly interested in developing end-to-end IR-based systems through representation learning. To date, several state-of-the-art reinforcement leaning-based methods [Qiu et al., 2020][Hildebrandt et al., 2020a] are proposed, making use of mutual information between semantic features of questions and KG's context, which are collected from interactions between agents and environments. Based on the KGQA framework in [Qiu et al., 2020], this work aims to investigate three research questions (RQs):

RQ1: Which entity linking method is appropriate for the KGQA task?

Starting nodes of KG reasoning are crucial to overall performance, because topic entities directly determines whether it is possible that an agent can arrive at target entities within limited steps. Though topic entities are labelled in questions explicitly in some datasets, models trained on them will lack the ability to generalize to raw and unprocessed questions.

RQ2: Does history-aware question representation matter in sequential decision making?

Given a question, the representation of words remains static at each reasoning step in the state-of-the-art model [Hildebrandt et al., 2020a], which omits the action history when computing the mutual information between queries and candidate answers. The work will investigate whether decision history provides useful insights in reasoning by introducing an attention module.

RQ3: Does reward shaping mechanism affect agent's performance?

In a policy-based RL procedure, a binary reward function is typically used to guide an agent towards right learning direction. The rewards are only generated after long reasoning episodes during which no feedback is available to agent. This results in weak supervision [Qiu et al., 2020] and low quality rewards [Lin et al., 2018]. The potential-based reward shaping mechanism [Ng et al., 1999] has been proven effective in addressing this problem. The work will evaluate the effect of shaping rewards in the following experiments.

The contributions of this thesis are threefold: i) an RL-based QA framework is proposed; ii) the model is tested on two benchmark datasets, which are **PathQuetion** [Zhou et al., 2018] and **MetaQA** [Zhang et al., 2018]; iii) extensive experiments are designed and conducted to investigate the aforementioned three research questions.

The rest of the thesis is organized as follows. Chapter 2 covers a holistic literature review regarding KGQA methodologies and relevant techniques. Chapter 3 is the main part that formally defines the KGQA problem and explains the proposed framework in detail. Chapter 4 exhibits experiment settings, empirical results, and an investigation on reasoning behaviors at different steps. Chapter 5 presents an interpretation on model's

performance and a limitation analysis. Finally, a conclusion is drawn in Chapter 6.

2

Related Work

This chapter summarizes related studies, presents a holistic review on the background, and elaborates how some of them have inspired this work. In the following, three topics are covered, which are: (i) the related KGQA methods, (ii) reinforcement learning and its applications on KGs, and (iii) a brief description on entity linking approaches.

2.1 Question Answering on Knowledge Graphs (KGQA)

KGQA aims to retrieve answers from knowledge graphs given natural language questions. Current mainstream approaches can be classified into two categories, which are semantic parsing-based (SP-based) methods and information retrieval-based (IR-based) methods.

2.1.1 Semantic Parsing-based Methods

Traditional Semantic Parsers

[Bast and Haussmann, 2015] proposes a system named *Aqqu* which automatically converts a natural language question to a SPARQL [1] query. The core thoughts are casting questions to one of three predefined templates and matching words in questions with KG relations. After constructing a set of query candidates, a ranking procedure is applied to identify the optimal matching one by taking 23 predefined ranking features into consideration, including features for entity matches, features for relation matches and other combined features.

Query Graphs

In [Reddy et al., 2014], the authors introduce *GraphParser* to treat semantic parsing as a graph matching problem. They first construct an ungrounded query graph out of the output of a combinatory categorical grammar (CCG) parser based on a given question and then map it to grounded subgraphs in Freebase [Bollacker et al., 2008]. The model is trained to identify the best grounded graph using weak supervision gained from denotations of natural language questions.

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

Another approach implementing query graphs in KGQA is *STAGG* [Yih et al., 2015]. Query graphs constructed under this framework can show high resemblance with KG subgraphs, and those query graphs can be mapped to lambda-calculus logical forms [Wong and Mooney, 2007]. Given a quetion, the whole solution includes three steps: identifying the question's topic entity, extracting the main relation between the topic entity and a possible answer entity, and enlarging the query graph based on other constraints described in that question. The three steps above also contribute as features captured for the purpose of KG and question match. The returned entity linking score is the first feature. Two convolutional neural networks (CNNs) are trained to indicate the quality in the core inferential chain: *PatChain*, making comparison between pattern and KG predicates, and *QuesEP*, comparing the question with the concatenation of topic entity and predicates. *STAGG* reduces the semantic parsing process to the construction of query graphs and the formation of staged problems.

A follow-up study [Bao et al., 2016] dedicates to extend the scalability of *STAGG* to questions including multiple constraints. The authors bring a Multi-Constraint Query Graph along with several types of constraints, e.g., entity constraint, type constraint, and explicit/implicit temporal constraint. Those constraints would be extracted from questions and added to query graphs of complex questions.

Encoder-Decoder Architectures

Some researchers find that encoder-decoder architectures can enable across-domain generalization without laborious human annotations. [Dong and Lapata, 2016] presents an attention-enhanced sequence-to-sequence model which maps input questions to logical form representations as depicted in 2.1. Furthermore, the authors upgrade the model to a sequence-to-tree framework to reflect the hierarchical structure of outputs.

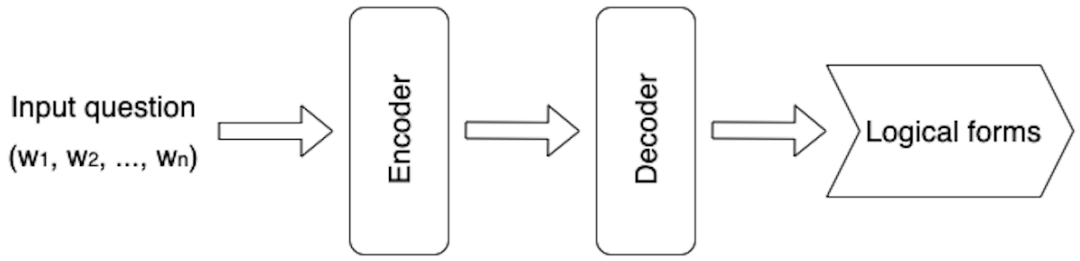

Figure 2.1: Input questions and logical representations are encoded and decoded end to end. [Dong and Lapata, 2016]

[Liang et al., 2016] proposes *Neural Symbolic Machine* which is a Manager-Programmer-Computer framework. The "programmer" converts language queries to programs as token sequences using an encoder-decoder model. The "computer" is responsible for

program execution based on a Lisp interpreter. And the "manager" collects rewards returned and provides a weak supervision during the training procedure.

One recent research [Jin et al., 2021] observes that previous studies on multi-hop question answering did not fully realize the impact of the relational chain order and relation types on performances, and neglected the implicit relations between topic entities and answers in KGs. Therefore, the authors proposed a *Relational Chain based Embedded KGQA* (Rce-KGQA) which learns explicit relation chains implied by questions and extracts implicit relation chain from KGs. The architecture includes an answering filtering module and a relational chain reasoning module. The former module aims to filter a set of candidate answers by ranking their scores concerning questions together with topic entities. A question semantic parser is implemented to provide vector representations of relations between topic entities and answer entities. The latter module is responsible for encoding shortest reasoning paths that lead to answer entities via a bidirectional long short-term memory (LSTM) and computing the similarity scores between relational reasoning chains and questions. This method utilizes intermediate relation chain signals in question answering which brings more reliable model reasoning.

In summary, semantic parsing-based approaches generally require deliberately designed templates and extracted features to convert questions into machine interpretable representations. Those methods can generate structured logical forms out of natural language questions at the cost of usability and accessibility to those don't possess linguistic knowledge. Another aspect is that it is challenging to train a neural semantic parser since gold logical forms are rare [Fu et al., 2020].

2.1.2 Information Retrieval-based Methods

This type KGQA methods follow two directions: feature engineering and representation learning. Generally speaking, an IR-based method starts with entity linking and finds answers in subgraphs around identified topic entities. Meanwhile, a scoring function is used to measure the semantic similarity between questions and candidates. This type of methods are capturing more and more attention for its template-free training process and the end-to-end QA workflow.

Feature Engineering-based Methods

[Yao and Van Durme, 2014] proposes a model that utilizes four features of input questions to construct *question feature graphs*, which consist of question words (e.g., what, where), question focuses (indicating answer types, e.g., name), question verbs (main verbs of questions, e.g., play), and question topics (topic entities retrieved by a named entity recognizer). A subset of entitiess within several hops from a topic entity is extracted as a topic graph, and it is combined with its question graph via a pairwise concatenation to capture association in their representations. From the observed aligned pairs, the co-occurrence matrix between words and KG relations is computed. The answer entity is judged by a binary classifier which takes question features and topic graphs as input.

Representation Learning Methods

Representation learning-based approaches leverage the neighborhood information of identified topic entities. All nodes within a certain subgraph are treated as candidate answers and ranked by outputs of a scoring function that measures the similarity between questions and candidates.

According to question types and reasoning procedures, they can be classified into two types: factoid QA and multi-hop QA.

Factoid Single-relation QA

Factoid single-relation QA task refers to a scenario that questions posed by users seek for matching factual triples in underlying KGs to obtain answers and no multi-hop reasoning over KGs is required. A recent SimpleQuestions dataset [Bordes et al., 2015] is commonly used as the benchmark in this thread of research.

[Yin et al., 2016] proposes a QA model that addresses single-relation factoid questions via a two-step pipeline. The first step is an entity linking task and the second one is a fact selection operation. Therefore, the authors introduce an architecture containing a character-level convolutional neural network (CNN) and a word-level CNN to solve these problems. At the first step, they utilize a BiLSTM-CRF (Bidirectional LSTM-Conditional Random Field) model to detect a question mention then do matching with entity candidates, and then they can find the topic entity of the question. As for the second step, patterns extracted from questions are compared with KG predicates via a Word-AMPCNN (Attentive MaxPooling Convolutional Neural Network), which adopts an attentive maxpooling mechanism during the pattern-predicate match procedure.

[Mohammed et al., 2017] conducts a set of experiments to understand the contribution of different neural network (NN) architectures in answering simple questions. Their work consists of three subtasks, which are entity linking, relation prediction and end-to-end QA based on SimpleQuestions dataset. Revealed by the experimental results, the effectiveness of some over complicated NN models is not necessarily superior than a baseline with simple heuristics. Even the baseline without NNs that combines logistic regression (LR) with conditional random field (CRF) model still acquires fairly good accuracy in entity linking for QA.

As proposed in [Sun et al., 2018], the model *GRAFT-Net* performs QA tasks by combining knowledge graphs with questions, which is an early-fusion strategy. To answer a natural language question, the model takes in a heterogeneous question graph formed from both KG facts and the text of queries, which allows multiple information sources to be considered. Therefore, the model is trained to classify whether node is answer entity in the question graph containing text documents of question and entities from KGs.

Multi-hop QA

As for complicated questions that cannot be solved within one-hop reasoning on KGs, methods have been raised to handle such multi-hop questions.

[Miller et al., 2016] proposes a *Key-Value Memory Network* (KV-MemNN) to assist in answering general questions either within or out of the domain of KG via reading documents directly. Because information in text can be far less structured than that in KB, KV-MemNN first stores facts in memory which could be retrieved by keys in

terms of questions. There are three steps involved in dealing with memory, key hashing (to filter those memories that share words with the input question), key addressing (to assign each candidate key a probability regarding the question) and value reading (to compute the weighted sum of memory values with their addressing probabilities). The framework allows prior knowledge to be stored in a structured way before reasoning, and it bridges the gap between knowledge graph and text.

Considering the encoding of differences between questions and KG components, a novel model is introduced in [Chen et al., 2019], namely *Bidirectional Attentive Memory networks* (BAMnet). There are four components in the model: an input module, a memory module, a reasoning module and an answer module. The input module encodes question embeddings via a bidirectional LSTM. The memory module evaluates each candidate around the topic entity through three types of information, which are answer type, answer path and answer context. After that, the aforementioned knowledge will be stored in a key-value memory module. In the reasoning module, a two bidirectional attention network is applied during the interaction between KGs and questions. The first network focuses on parts of a question given a KG and the components of a KG regarding a question, and the secondary one aims to enhance representations of KGs and questions. Unlike methods that encode questions and KGs separately, BAMnet is capable of capturing two-way interactions between them, and the usage of attention mechanism also brings better interpretability.

[Qiu et al., 2020] puts forward a stepwise multi-hop reasoning model based on reinforcement learning (RL). Input natural language questions are encoded by a bidirectional gated recurrent unit (GRU) network, and an agent can traverse over KGs in a sequential decision procedure, judging which action to take next. At each step, representations of question are updated according to the present action spaces. A history encoder is also implemented to incorporate actions taken in agent's paths. Besides, the policy-based RL agent takes not only the final reward after one reasoning episode, it can be guided by the intermediate rewards during transitions from states to states.

In summary, IR-based methods do not require hand-crafted rules and annotations which make model training expensive, but they are more concerned about feature similarities between KG nodes and the questions. IR-based models can be efficiently trained in an end-to-end fashion, but some of them might be unable to well handle rather complex questions with multiple constraints.

2.1.3 Visual Question Answering (VQA) over KGs

VQA is a task that retrieves answers from images with respect to given questions. It has been noticed that KGQA and VQA share common properties. They are obliged to answer natural language questions, and KGQA often considers the semantic relational context of KGs while VQA must use spatial context to locate the target objects implied by questions. Therefore, researchers make a step towards enhancing VQA performance by incorporating the advancements in KGQA.

One work among those is introduced in [Hildebrandt et al., 2020a]. It constitutes VQA as a path-finding problem over scene graphs and trains an agent based on reinforcement

learning to conduct random-walks to form reasoning paths leading to targets. The model embeds KG relations and entities via a graph-attention-network (GAT) through which the embeddings of entities are influenced by their neighbors in the graph, and generates question vectors via a Transformer [Vaswani et al., 2017]. During the reasoning process, an action taken at a time is encoded into a history encoder, and provides guidance to the selection of next actions. In the end of a path, the agent receives a 0/1 reward depending on whether answers are correct.

2.1.4 Entity Linking

Entity linking (EL) is an important step in KGQA, which correctly identifies the mention in a question and link it to an entity in the KG [Sevgili et al., 2020]. Researches on KGQA cannot circumvent this problem easily since its a preliminary stage before starting a reasoning over a KG. For instance, in a natural language question "*what is the place of birth of Tesla?*", the mention "*Tesla*" is supposed to be linked with entity "*Nicola Tesla*" the physicist with a higher probability than with entity "*Tesla, Inc*" the company. According to the order of mention detection, there are two types of entity linking methods: passive entity linker and active entity linker [Yin et al., 2016].

Passive Entity Linker

A passive entity linker refers to a mechanism that first search a set of candidate entities based on question words and use candidates to detect the mention in question.

In [Yin et al., 2016], the authors introduce a simple passive EL solution, which takes three factors to rank candidates based on strings. Though this method does not consider semantic information, it is shown that the three factors are also necessary for a fairly good entity linker. At first, the passive entity linker derive the longest consecutive common subsequence (LCCS) between the natural language query and entities in the KG and form a set of candidate entities that have a none-zero LCCS with the question. Then three key metrics are defined, and they will be combined into a scoring function to rank candidate entities. This approach is proved to be effective and easy to use in simple question answering tasks.

Active Entity Linker

An active entity linker first retrieves the mention in the question, then finds candidates given the mention span.

Inspired by [Golub and He, 2016], an active entity linking mechanism is proposed by [Yin et al., 2016], which is a bidirectional LSTM-CRF model used to identify mention in a question by sequential labeling. Afterwards, the detected mention is served as a reference to search for entities from knowledge graph. Different from passive entity linker, this kind of method only generates one mention for a question, and does not rely on returned candidate entities. Nevertheless, it requires questions labeled with topic entities for training model.

Neural Entity Linker

Recently, a class of neural approaches to solve the entity linking problem have been developed, aiming to reduce burden of hand-crafted feature engineering and to bring improvements on the EL task.

[Ganea and Hofmann, 2017] present a attentive neural model which uses local context window, combines word and entity embeddings and enables mentions in a query to be jointly resolved by a conditional random field (CRF). The whole system is differentiable for inferring ambiguous entities.

An end-to-end neural model based passive entity linker is proposed in the work of [Kolitsas et al., 2018], which can conditions the semantic similarity between mentions and the strongest context support of its most matching candidate entity. In this approach, mention detection and entity disambiguation are conducted simultaneously by the neural module, in other words, if the mention has at least one candidate entity, it can still be judged as invalid and unlinkable to entities in the KG by the neural network.

2.2 Reinforcement Learning (RL)

Reinforcement learning (RL) learns how to map environment to action and to maximize the numerical reward signal. RL considers a problem of a series of goal-directed agent interactions with the environment [Sutton and Barto, 2018]. This inherent attribute enables RL methodology to be adapted to a scenario of multi-hop KGQA, in which there is a knowledge graph serving as the environment and the potential answer can be retrieved in a sequential decision process by the RL agent. Such approaches bring better interpretability in model behavior compared to other information retrieval-based methods since all reasoning steps are explicitly recorded.

2.2.1 RL in Path Finding

[Xiong et al., 2017] first proposed a RL-based fact-prediction algorithm, namely *DeepPath*. For rules to be retrieved from KG, it trains a RL model to perform the reasoning. There are two stages including pre-training and re-training on models, the former is a supervised learning which tries to maximize the expected cumulative reward in using BFS algorithm to find correct paths between entities, and the latter performs the RL training procedure in order to solve the path-ranking problem.

Afterwards, [Das et al., 2017] combine RL mechanism into a novel algorithm, *MINERVA*, to address more complex link prediction problem, in which only start entity and relation are provided, while the target entity is missing. The model involves a LSTM-based history encoder and a two-layer policy network which maps the combined information of history, query and observation to a probability distribution of actions.

Instead of implementing one RL agent, [Hildebrandt et al., 2020b] presents an interesting debate system *R2D2* which has two RL agents and one judge, the two agents are

trained to constitute an argument and path to favor a thesis or antithesis, and the judge works as a binary classifier to evaluate the argument extracted by two agents.

[Wang et al., 2019] summarizes techniques in previous work and brings graph attention mechanism (GAT) into their RL-based path reasoning algorithm, *AttnPath*, therefore the agent pays more attention to highly relevant relations and neighbors in selecting actions. Considering the drawbacks in their previous work, the authors further improve their model by incorporating a memory module in [Li et al., 2021] and apply several tricks, including action dropout, reward shaping and force forward.

2.2.2 RL in KGQA

In addition to the SRN algorithm introduced in section 2.1.2 that formulates multi-relation QA as a Markov decision process (MDP) via RL, progress has been made in the extended KGQA field. Different from question answering problems involving single question answer pair, [Kaiser et al., 2021] present a RL model *CONQUER* to deal with question answering in conversations over KG. Questions in a conversation context can be incomplete by itself since it may use information from previous utterances, therefore users may rephrase their questions as reformulations if the answer to the last question is incorrect until a correct answer is returned by the system. Given the current question and previous questions, the model keeps a set of relevant entities based on the whole conversation. Starting from these entities in turn, all end nodes reached by the RL agent after a reasoning walk over KG will be candidates for this turn and be aggregated to generate a final response. The policy network is trained by noisy rewards from the reformulation likelihood provided by a BERT predictor.

2.3 Summary

KGQA has been exhaustively studied over the past few years, forming two main branches including semantic parsing-based methods and information retrieval-based methods.

The SP-based approaches can handle complex questions with deliberately designed templates and rules, parse natural language queries into machine interpretable logical forms. They can be divided into the traditional type and the neural-based type [Fu et al., 2020]. The traditional ones rely on hand-crafeted tempaltes and require researchers to be rather familiar with linguistic knowledge. The neural based SP methods can construct parsers via neural models to enhance scalability, but it is still a challenge to train a semantic parser due to its demand for a great amount of annotations and the lack of gold logical forms.

The IR-based methods are more flexible in training compared with SP-based approaches. They locate the topic entity in the KG and define a subgraph around it to search for the answer node. The semantic information extracted from the subgraph and the question are combined to capture their relevance, and find the target entity. Most of the IR approaches are not well interpretable, whereas the RL-based methods form

the reasoning paths over KG step by step, and each decision is supported by collected feature in the environment and queries.

One most important part in KGQA is entity linking, which identifies mentions in the question and retrieves the most relevant node from the KG as the topic entity. Especially for those reasoning models, a correct starting point is the preliminary for a correct path. Mainstream approaches are classified into passive methods sand active ones according to the order of mention detection. Passive entity linkers first segment questions into spans and find all relevant candidate and rank them using a scoring function. While active entity linkers determine the mention in a question first and then return candidates for the mention.

The methodology developed in this project is motivated by reinforcement learning mechanism, which will be illustrated in the next section.

3

Methodology

In this section, the methodology and architecture design are described in detail.

3.1 Background and Problem Definition

This section introduces background knowledge relating to this work, including the formal notation of a knowledge graph, and the definition of KGQA tasks. From there, the proposed methodology is explained in detail.

A knowledge graph (KG) is a set of triples, let $G = \{(e_s, r, e_o) \mid e_s, e_o \in E, r \in R\}$ denotes a knowledge graph, where E is the set of entities and R is the set of relations in the KG. (e_s, r, e_o) represents a triple of subject-relation-object data entry, where the subject and object come from E and the relation comes from R. A valid fact means that for a subject node e_s and an object node e_o, there exists one relation r that connects these two entities in the KG.

A knowledge graph is not necessarily complete per se in most cases due to missing information during construction, and this feature strongly motivates the advancement of studies in path reasoning and relation predicting as introduced above. The task of *multi-hop knowledge graph question answering* cannot circumvent this problem neither. It would be ideal if a question can be answered by a triple from the KG. Even in a limited specific domain of knowledge, however, the variety of natural language queries cannot be enumerated, and it is often an impossible mission to answer a posed question via a single triple. Therefore, learning to reason on multiple adjacent triples over KG to form a proper answer is a promising research direction.

Given a knowledge graph G, and a natural language query $Q = (w_1, w_2, ..., w_n)$, which is composed of n words, for an identified topic entity e_s according to the question, some relations in R will be the best description for predicates in the question, and there exists a retrieved answer entity e_o that can be reached by a path starting from e_s up to a restricted length, and the length is equal to the minimal number of hops between these two nodes if the edges are not weighted.

3.2 Proposed Framework

Overview. Inspired by previous researches in the field of KGQA, the proposed novel framework is based on reinforcement learning, and the objective is dealing with multi-hop question answering task on KGs.

Different from workflows in [Qiu et al., 2020], this work starts resolving a natural language question from entity linking first. In real-world applications, topic entities may not be explicitly labelled before reasoning, so it is one critical step to extract topic entities from given questions.

In addition, attention mechanism is usually applied to learn contextual information in previous researches. Specifically, the *SRN* model in [Qiu et al., 2020] utilizes action candidates to update embedding representations of questions at each step via an attention between relations and question words, and generates *relation-aware* question representations, while the visual question answering model in [Hildebrandt et al., 2020a] directly passes initialized entity embeddings of scene graphs to a Graph Attention Network (GAT) with a self-attention mechanism to capture context information from neighbors. In this framework, question answering is considered as an information retrieval process in which question is resolved in steps. The decision history records which part of the question has been covered, and the next step is supposed to pay more attention to those question parts that have not been exploited according to the history.

In general cases, an RL agent learns from reward signals returned by a default 0/1 reward function after taking final actions of reasoning paths, and tries to maximize the expected rewards. The drawback is that no intermediate feedback is available, so the reward shaping technique is proposed to provide more supervision to the agent based on potentials of states. RL agent training can benefit from such immediate rewards compared to sparse terminal rewards, and this framework adopts a potential-based reward function to supply additional state information.

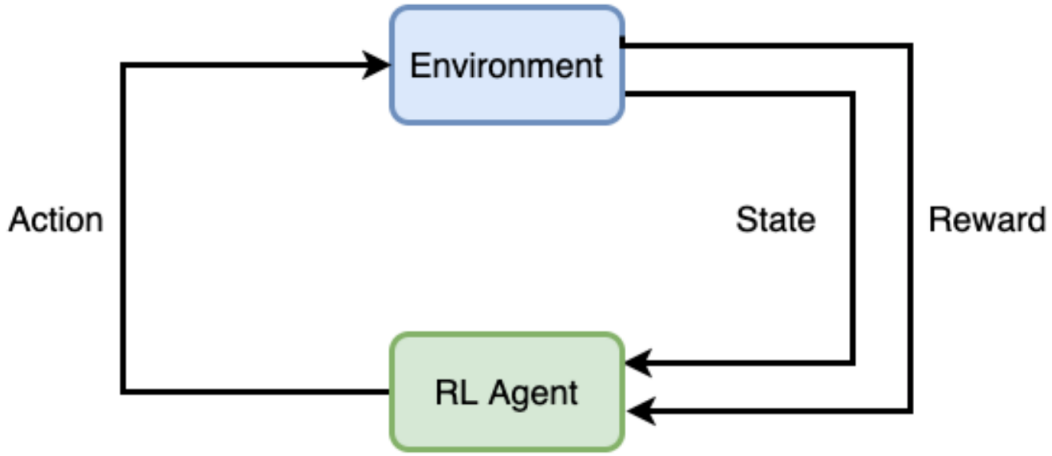

Figure 3.1: The agent-environment interactions

3.2.1 Reinforcement Learning Formulation

The RL system includes two main components: one is the **environment** that provides the structure and semantic information of knowledge graphs, and the other is the **agent** trained to learn from reward signals using the REINFORCE algorithm [Williams, 1992]. In a word, the functionalities of the framework are implemented by the agent which interacts with and receives feedback from its environment to make decisions and take actions.

The interactions can be interpreted as a finite Markov decision process (MDP) as depicted in Figure 3.1, in which a finite set of states S, actions A and rewards R form a sequential trajectory [Sutton and Barto, 2018]:

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, ... \tag{3.1}$$

State. The states in the environment represent the agent's status at different time steps, and encode the query, the topic entity, the current location and the trajectory history. In the proposed RL framework, a state at step t is formulated as follows:

$$S_t = (q, e_s, e_t, h_t) \tag{3.2}$$

where q is the input query, e_s is the starting node, e_t reflects the entity on which the agent locates at time t, and h_t encodes the previous decisions in the trajectory.
Action. When the agent arrives at an entity e_t, it may face a set of action candidates A_{S_t} composed of outgoing relation-entity pairs:

$$A_{S_t} = \{(r, e) \mid r \in R, e \in E, (e_t, r, e) \in G\} \tag{3.3}$$

Since the original data sets only contain triples that allow the agent to move along the same direction as indicated, reverse relations of original ones are added between entities to bring more flexibility in reasoning, i.e., for a fact triple (e_1, r, e_2) in the knowledge base, a reverse triple is generated as (e_2, r^{-1}, e_1) when constructing the KG. For example, the relation *written_by* has an inverse counterpart $written\_by^{-1}$ which allows bidirectional connections between two entities [Xiong et al., 2017] [Das et al., 2017].
Transition. Upon taking an action, the system updates current state via transition. A transition function between two states is defined as

$$f : S_t \times r_t \mapsto S_{t+1}. \tag{3.4}$$

Reward. A reward signal indicates the objective of a reinforcement learning task. Therefore, an RL agent is learnt by maximizing the expected reward after a run over a KG. In a default policy-based RL process, the agent receives a terminal reward according to a binary reward function,

$$R = \begin{cases} 1, \text{ if } e_t = e_{target}, \\ 0, \text{ otherwise}, \end{cases} \tag{3.5}$$

where e_t is the current entity at which the agent locates, and e_{target} is one answer entity for the query.

3.2.2 Policy Network

This section explains implementation details of the policy network.

Embeddings. Entities and relations of the knowledge graph G are initialized by pre-trained vector embeddings based on TransE, whose scoring function is shown below [Bordes et al., 2013]:

$$f_{\mathbf{r}}(\mathbf{h}, \mathbf{t}) = -\| \mathbf{h} + \mathbf{r} - \mathbf{t} \|_{1/2}, \tag{3.6}$$

where $\mathbf{h}$ and $\mathbf{t}$ denote the embeddings of head entity and tail entity respectively.

As for input natural language questions, words are represented by GloVe[1] pretrained word embeddings [Pennington et al., 2014]. Words out of the GloVe vocabulary are assigned with an average of all word vectors as suggested by the author[2]. The initialized embeddings as well as the input of the policy network are vectors in R^d.

Policy. The policy defined over states is a function $\pi_\theta : S \mapsto A$ that maps state S to action A [Ng et al., 1999], where θ is the learned parameter of the policy. To provide necessary state information, the network is composed of several modules as depicted in Figure 3.2, including a question encoder, a history encoder, an attention module, and a perceptron for decision making.

Given a question of length n, $Q = (w_1, w_2, ..., w_n)$, the initialized word vectors are encoded by a bidirectional GRU (BiGRU) network sequentially. The mention of a topic entity is extracted by the entity linker introduced in section 3.2.3 and is replaced by a symbol "$\langle e \rangle$" to let policy network focus on the predicates in the question. The BiGRU network generates a context aware representation of the question in the form $\boldsymbol{Q} = [\boldsymbol{w_1}, \boldsymbol{w_2}, ..., \boldsymbol{w_m}] \in \mathsf{R}^{d \ \ m}$, where m is the length of the sequence after replaceing the topic entity's mention with "$\langle e \rangle$", and $\boldsymbol{w_i}$ is a word vector that consists of forward and backward outputs from the BiGRU network.

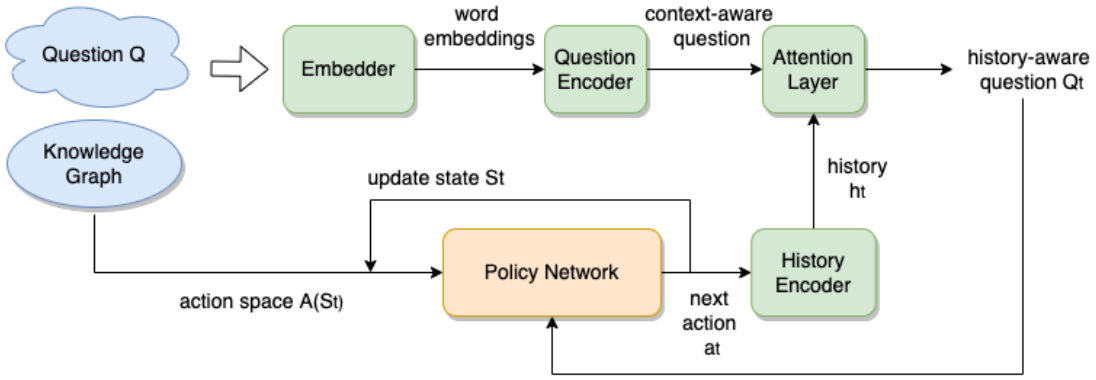

Figure 3.2: The overall model architecture

With an identified topic entity e_s, the agent starts from an initial state $S_0 = (\boldsymbol{Q}, e_s, e_s, \boldsymbol{h_0})$, where the current entity is also located at the starting node, and a history h_0 is updated

[1] *https://nlp.stanford.edu/projects/glove/*
[2] *https://groups.google.com/g/globalvectors/c/9w8ZADXJclA/m/hRdn4prm-XUJ*

via a history encoder in further steps. The history encoder is based on an LSTM network, and the history h_t is updated with previous history $h_{t\ 1}$ and last action's relation taken by the agent r_t,

$$\boldsymbol{h}_t = LSTM(\boldsymbol{h}_{t\ 1}, \boldsymbol{r}_{t\ 1}) \in \mathsf{R}^d, \tag{3.7}$$

where t starts from 1, $\boldsymbol{h}_0$ and $\boldsymbol{r}_0$ are initialized by zero vectors.

Having the encoded question and history, the attention layer learns a set of attention weights over question words. This operation enables the policy network to pay more attention to parts of the question in light of history. For a question $\boldsymbol{Q} = (\boldsymbol{w_1}, \boldsymbol{w_2}, ..., \boldsymbol{w_m}) \in \mathsf{R}^{d\ m}$, and a history $\boldsymbol{h}_{t\ 1}$, the element-wise production of these two vectors is fed to the attention module followed by a Softmax layer. The output will be a vector of attention scores for words $\boldsymbol{w_i}, i \in 1, 2, ..., m$. By computing the weighted sum of question words, a history aware question representation $\boldsymbol{Q_t}$ is generated for producing a probability distribution over candidate actions.

$$\boldsymbol{Q}_t = \sum_{i=1}^{m} \alpha_{i,t} \cdot \boldsymbol{w}_i, \tag{3.8}$$

$$\alpha_{i,t} = Softmax(\beta_{i,t}), \tag{3.9}$$

$$\beta_{i,t} = \boldsymbol{W_{Attn}} \cdot (\boldsymbol{h}_{t\ 1} \odot \boldsymbol{w}_i) + b_{Attn}, \tag{3.10}$$

where $\boldsymbol{w}_i$ is the i-th word's embedding, $\boldsymbol{W_{Attn}} \in \mathsf{R}^d$ and $b_{Attn} \in \mathsf{R}$ are the weights and bias of the attention layer, $\alpha_{i,t} \in \mathsf{R}$ is the attention score of $\boldsymbol{w}_i$.

Finally, a two-layer perceptron is used as the policy module predicting a probability distribution given the current state S_t, the history-aware question representation $\boldsymbol{Q_t}$ together with the action space A_{S_t}, and an action sampled from the distribution will be absorbed into the trajectory history and used to update the state to S_{t+1}. To be more specific, the module calculates the semantic similarity between actions and the concatenation of history $\boldsymbol{h}_t$ and question $\boldsymbol{Q}_t$ through passing them into the attention layers, whose first layer is followed by a ReLU activation function, and feed the outputs to a Softmax layer, the distribution is computed as follows:

$$P(A_{S_t}) = Softmax(\boldsymbol{W_2} \cdot ReLU(\boldsymbol{W_1} \cdot [\boldsymbol{h}_t;\ \boldsymbol{Q}_t]) \cdot A_{S_t}), \tag{3.11}$$

where $\boldsymbol{W_1}$ and $\boldsymbol{W_2}$ are the weights of two perceptron layers, and A_{S_t} denotes the action space of the state S_t.

3.2.3 Entity Linker

For the question answering task, entity linking is the step that detects mentions from questions and matches with entities in the knowledge graph. Depending on the order of mention detection and candidates search, the entity linker methods are divided into two classes, passive EL and active EL. Considering the consistency of mentions and entity

names of used dataset, a **passive entity linking** technique is employed as presented in [Yin et al., 2016].

The preliminary is to derive the longest consecutive common subsequence (LCCS) in word level between the input query and KG entities. For instance, given a question "when was the New York University founded?" and an entity "New York State", their LCCS is the sequence "new york" of length 2. The EL procedure is described in the following.

Step 1: The input question sequence and KG entity names are split into tokens and converted to their lowercase for matching.

Step 2: For each entity, the algorithm is applied to calculate the longest consecutive common subsequence σ between it and the question, and all those have a non-empty LCCS are gathered into a set of candidate topic entities C_e.

Step 3: Regarding a candidate entity e, three factors are extracted for computing its score, which are $a =\mid \sigma \mid / \mid Q \mid$, $b =\mid \sigma \mid / \mid e \mid$, and $c = p/ \mid Q \mid$, where $\mid \cdot \mid$ denotes the sequence length of question or entity, p is the last token's location in the question Q.

Step 4: Ranking all candidates in $e \in C_e$ by the function $score(e) = \alpha a+\beta b+(1-\alpha-\beta)c$, where α, β, $1-\alpha-\beta$ are the weights of three parameters.

In the above score function, the parameter a favors candidates that cover more tokens in the question, parameter b prefers candidates having LCCS accounting for larger parts of themselves, and parameter c supports candidates show up close to the end of question Q, which coincides with a general phenomenon that mentions tend to be located at positions far from the beginning.

The top ranked entity is considered to be the topic entity in terms of the question, and the longest consecutive common subsequence is the identified mention for Q. The QA datasets are preprocessed with this method before being fed into the policy network, and it is proved that this algorithm works effectively in resolving the entity linking problem here.

3.2.4 Reward Shaping

During the sequential decision process, the objective of the QA task is reflected by the reward function, and the agent receives a terminal reward based on its final state. The optimal policy is determined given a reward function and a policy model of a domain, and a potential-based reward shaping function does not affect the policy invariance [Ng et al., 1999].

This type of shaping rewards are supposed to be derived from state transitions, in which each state is associated with a value of a potential function $\Phi(S_t)$. As illustrated in [Ng et al., 1999], a general form of shaping rewards $R^{\emptyset}$ looks like

$$R^{\emptyset} = R + F, \tag{3.12}$$

$$F(S_t, r_t, S_{t+1}) = \gamma\Phi(S_{t+1}) - \Phi(S_t), \tag{3.13}$$

where γ is a discount factor, $\Phi(\cdot)$ is a function that quantifies states with numeric values, r_t is the relation of action $a_t = (r_t, e_t)$, F is the potential-based shaping function, and R is the default binary reward function. In this framework, the potential function is defined as

$$\Phi(S_t) = \begin{cases} ReLU(cos(Q_t, h_t)) & \text{if } t > 1, \\ 0 & \text{if } t = 1, \end{cases} \tag{3.14}$$

where cosine similarity between history-aware question Q_t and history h_t is passed through a ReLU layer.

Different from terminal rewards, the shaping rewards bring intermediate feedbacks to RL processes. In addition to potential-based function, there are also other reward functions, such as multiple reward criteria depicted in [Xiong et al., 2017], which includes three factors: whether the agent reaches the target (global accuracy), the inverse of the path length (path efficiency), and how much current path resembles previous ones (path diversity).

3.3 Model Training

In practice, the parameters of the policy network π_θ are learned through maximizing the expected cumulative reward in each episode,

$$J(\theta) = \mathsf{E}_D[\mathsf{E}_{A_1,A_2,\ldots,A_{T-1}} \ _\pi[\sum_{t=1}^{T} \gamma^t \ {}^1R'(S_t, a_t, S_{t+1})\pi_\theta(a_t \mid S_t)]], \tag{3.15}$$

where D is the dataset of question-answer pairs, A_i and a_t are the action space and the action taken at step t respectively, and γ is the discount factor as used in equation 3.13. The expectation is approximated by empirical average results over training samples. The objective function is optimized by the REINFORCE algorithm [Williams, 1992], which uses the complete returned rewards of the whole episode [Sutton and Barto, 2018], and works as a Markov decision process.

Algorithm 1 Natural language question answering via knowledge graph reasoning

Require: Knowledge graph $G = (E, R)$; question Q, denote $\{w_i\}_{i=1}^{m}$; max steps T; discount factor η.

Ensure: The predicted sequential trajectory of states and actions maximize the expected cumulative rewards.

Apply the entity linker to the question Q and extract a topic entity e_s in G
Initialize the KG components and the question with embeddings
$t \leftarrow 1, e_t \leftarrow e_s$
$\boldsymbol{h_0} \leftarrow \mathbf{0}, \boldsymbol{r_0} \leftarrow \mathbf{0}, \boldsymbol{h_t} \leftarrow GRU(\boldsymbol{h_0})$
$\boldsymbol{Q} \leftarrow BiGRU(Q)$
Assign uniform weights to tokens, $\{\alpha_{i,0}\}_{i=1}^{m} \leftarrow \frac{1}{m}$
$S_1 \leftarrow (\boldsymbol{Q}, e_s, e_t, \boldsymbol{h_t})$
$R^{'} \leftarrow 0$
while $t \leq T$ **do**
    $\boldsymbol{\beta_t} \leftarrow \boldsymbol{W_{Attn}} \cdot (\boldsymbol{h_t} \odot \boldsymbol{Q}) + b_{Attn}$
    $\boldsymbol{\alpha_t} \leftarrow Softmax(\boldsymbol{\beta_t})$
    $\boldsymbol{Q_t} \leftarrow \sum_{i=1}^{m} \alpha_{i,t} \boldsymbol{w_i}$
    Retrieve the action space A_{S_t} for current entity e_t
    for $a_t = (\boldsymbol{r_t}, e_{t+1})$ in A_{S_t} **do**
        $S(a_t) \leftarrow \boldsymbol{r_t} \cdot \boldsymbol{W_2} \cdot ReLU(\boldsymbol{W_1} \cdot [\boldsymbol{h_t} : \boldsymbol{Q_t}])$
        $P(a_t) \leftarrow Softmax(S(a_t))$
    end for
    Sample action $(\boldsymbol{r_t}, e_{t+1})$ from the distribution $P(A_{S_t})$
    $\boldsymbol{h_{t+1}} \leftarrow GRU(\boldsymbol{h_t}, \boldsymbol{r_t})$
    $S_{t+1} \leftarrow (\boldsymbol{Q}, e_s, e_{t+1}, \boldsymbol{h_{t+1}})$
    $R \leftarrow \mathbb{I}\{e_t \text{ is an answer entity}\}$
    $F \leftarrow \gamma\Phi(S_{t+1}) - \Phi(S_t)$
    Calculate cumulative reward $R^{'} \leftarrow R^{'} + \eta^{t}\ {}^{1}(R + F)$
    $t \leftarrow t + 1$
end while

4

Experiments

This section presents the empirical implementation of the proposed framework on KGQA datasets from various domains.

4.1 Datasets

Table 4.1 shows the statistics of two benchmarks on which the experiments were conducted.

PathQuestion is a QA dataset first proposd by [Zhou et al., 2018], in which questions are created with templates. To construct PathQuestion, a subset of Freebase [Bollacker et al., 2008], namely FB13, is adopted to extract 2-hop and 3-hop paths. Those paths are used to generate corresponding natural language questions via handcrafted templates. In addition to question-answer pairs and the knowledge base, PathQuestions also provides a reference path for a given QA sample, which allows us to compare the predicted path with the groundtruth.

MetaQA is a film domain dataset constructed by [Zhang et al., 2018], including 1-hop, 2-hop and 3-hop training datasets. The 1-hop is directly derived from the original WikiMovies [1], which is composed of QA pairs and a knowledge base [Miller et al., 2016], and the 2-hop and 3-hop datasets are constructed by randomly sampling from a collection of templates.

Dataset	#Entities	#Relations	#Triples	# Questions
PathQuestion	2215	14	4049	7106
MetaQA	43234	9	134741	407513

Table 4.1: Statistics of QA datasets used in experiments

[1] Available at *https://research.fb.com/downloads/babi*

4.2 Compared Models

To evaluate the performance of this model, there are several models to be compared with, including the state-of-the-art SP-based methods and IR-based methods:

- **SRN** [Qiu et al., 2020] is an IR-based model dealing with natural language question answering tasks over KG. The proposed method initiates an RL reasoning chain from a known topic entity and terminates when the maximum length is reached.
- **Rce-KGQA** [Jin et al., 2021] is an SP-based SOTA architecture, which combines the explicit semantic relational chain in a question and implicit relational chain in the KG, and utilizes a relational chain reasoning module to prune candidate entities.
- **MemN2N** [Sukhbaatar et al., 2015] is a memory network that stores all triples in the memory units, which converts the memory and the questions into vectors to compute their similarities. Moreover, MemN2N can handle with multiple-hop operations via stacking memory layers.

4.3 Implementation Details

KG Initialization
Similar to previous approaches [Xiong et al., 2017], KGs are augmented by updating the uni-directional relations between nodes with bidirectional relations. For a triple (e_s, r, e_o) existing in the knowledge graph, a reverse triple (e_o, r^{-1}, e_s) is added to the graph correspondingly. The KG components are initialized by pre-trained TransE embeddings following the OpenKE framework[2], with the training times = 500, number of batches = 100, and the output embedding dimension $d = 100$.

QA Data Preprocessing
The preprocessing of QA data is two-fold, first applying entity linking to plain natural language questions, which returns a serial of question words and a topic entity, then converting the tokenized question to a vector of GloVe embeddings in word level.

Since the benchmarks used in experiments have mentions explicitly labeled in questions, this work eliminates the symbols that specify mentions and tokenize question sequences to lists of words, and all tokens are in lowercase, without lemmatization or stemming. Afterwards, the processed questions and the KG components are fed into the passive entity linker described in section 3.2.3. For each question, only one mention-topic entity pair is kept according to score ranking of candidate entities.

Given the question sequence, words are mapped to GloVe embeddings of dimension 100. As for words not included in the GloVe dataset (out-of-vocabulary), an average over all words vectors is used instead.

[2]Available at *https://github.com/thunlp/OpenKE*

Hyperparameters
By conducting cross-validation over datasets, the model architecture's hyperparameters are initialized as follows:

The question encoder is a bidirectional GRU network including 2 layers, and the hidden dimension is set to 50. The history encoder is a 4-layer GRU network, with a $hidden\ dimension = 100$. A $dropout\ rate = 0.2$ is applied to all GRU layers. The attention module is a single-layer perceptron followed by a Softmax. The policy module is a two-layer preceptron, with a hidden dimension $d = 100$, and the output is passed through a Softmax as well. For different subsets of training data, the maximum step is set to their expected hops, e.g., $max\ step = 2$ for PQ 2-hop and MetaQA 2-hop. The model is optimized by the Adam optimizer during training process with an initial learning rate $lr = 1e - 4$.

4.4 Experimental Results and Analyses

Before conducting the experiments, the QA datasets are randomly split into three subsets, with a portion of $train : validation : test = 8 : 1 : 1$. Each combination of dataset and model settings is evaluated with experiments for at least 10 complete training processes, and each process is trained until its performance has not been improved for more than 5 consecutive epochs. The trainer traverse over the complete training set in each epoch. Since the reference answers are not necessarily unique, that is, a question can have a answer set $\{e_{ans}\}$ containing more than one target entities. To evaluate the model's performance, the metric **hits@1** is used to count the ratio of samples that has a predicted target belonging to its answer entities in the test dataset.

Table 4.2 and 4.3 show hits@1 results on two benchmarks.

Model	PQ 2-hop	PQ 3-hop
Random	15.1	10.4
MemN2N (2015)	89.5	79.2
SRN (2020)	**96.3**	**89.2**
This Model	55.82	58.95

Table 4.2: Experimental results on two subsets of PathQuestion (% hits@1)

From the listed results on PathQuestion compared with three baselines and the random case (no model is implemented and the predictions are made randomly), this model performs much better than the random results, though lacks for competency when facing the present state-of-the-art methods.

As for the MetaQA dataset, the proposed model outperforms the random case and the MemN2N by a big margin in 1-hop and 2-hop subsets, but still falls behind the SRN

Model	MetaQA 1-hop	MetaQA 2-hop	MetaQA 3-hop
Random	13.1	9.8	1.3
MemN2N (2015)	78.5	30.5	19.0
SRN (2020)	97.0	95.1	75.2
Rce-KGQA (2021)	**98.3**	**99.7**	**97.9**
This Model	95.61	48.54	16.71

Table 4.3: Experimental results on three subsets of MetaQA dataset (% hits@1)

and the Rce-KGQA model especially in multi-hop tasks. That being said, the SRN has a performance slightly superior than this model in the MetaQA 1-hop task.

The final results in the two tables can only provide a general performance of the RL agent, lacking of deeper insights on reasoning behaviors. Therefore the statistics of intermediate reasoning steps based on the validation dataset are further examined. To achieve this goal, reference paths are in need for comparison with the predicted paths. The PathQuestion dataset is consequently the most suitable choice for its additional attribute "*path*", which are strings in the form "*topic_entity#relation1#entity2#...#answer_entity*". In experiments, the relation names are extracted from the paths, and results of statistics are presented in the following (from Figure 4.1 to Figure 4.5).

It is noteworthy that the reference paths are **not necessarily the only right paths** for the natural language questions in the dataset, one reason is that there might be multiple answer entities, and the other one is that a topic entity and its answer entity can be connected by more than one path in some cases. The probabilities in Figure 4.1 and Figure 4.2 illustrate the behaviors of this model in comparison with the "gold" reference path.

As shown in Figure 4.1, there are two groups represented by green and blue bars, denoting the conditional probabilities of current prediction's correctness. The green bars are probabilities conditioned on that the previous action is correctly predicted, i.e., P(current step is correct | previous step is correct) and P(current step is not correct | previous step is correct), while the blue bars are probabilities conditioned on a wrong previous step. The term "**correct**" here means that the prediction of an action coincides with the reference action at that step, and the term "**wrong**" indicates a divergence between the prediction and the reference. Moreover, different relation types are not distinguished in counting.

In Figure 4.2, a similar pattern is observed on probabilities conditioned on step 1 as in Figure 4.1, meaning that a false action taken in step 1 results in a rather high likelihood of a false prediction in step 2 (around 0.97). However, this is not the case for conditional probabilities given step 2's results in PQ 3-hop dataset. Correct predictions in step 2 enhance the possibility of correct subsequent actions, but wrong predictions in step 2,

however, do not substantially weaken the correct ratio in step 3 as also shown in Figure 4.3.

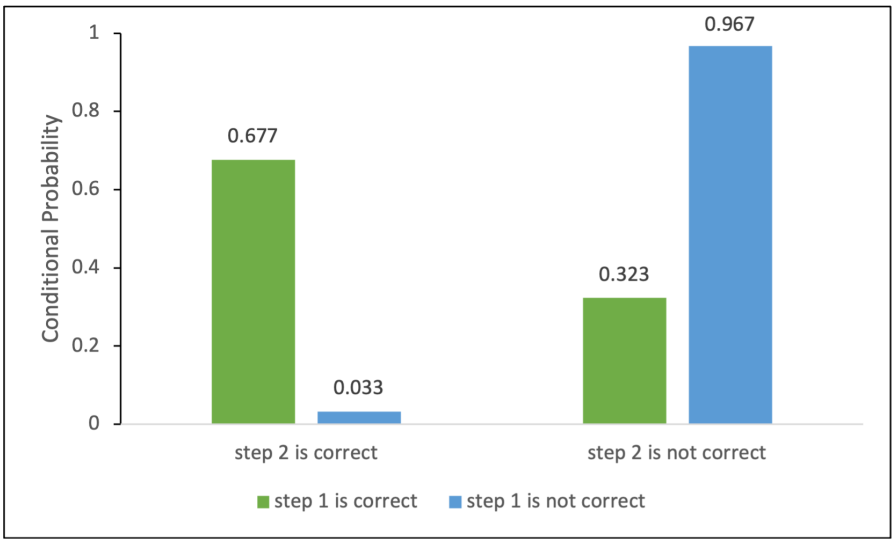

Figure 4.1: Conditional probability of prediction behaviors on PQ 2-hop dataset

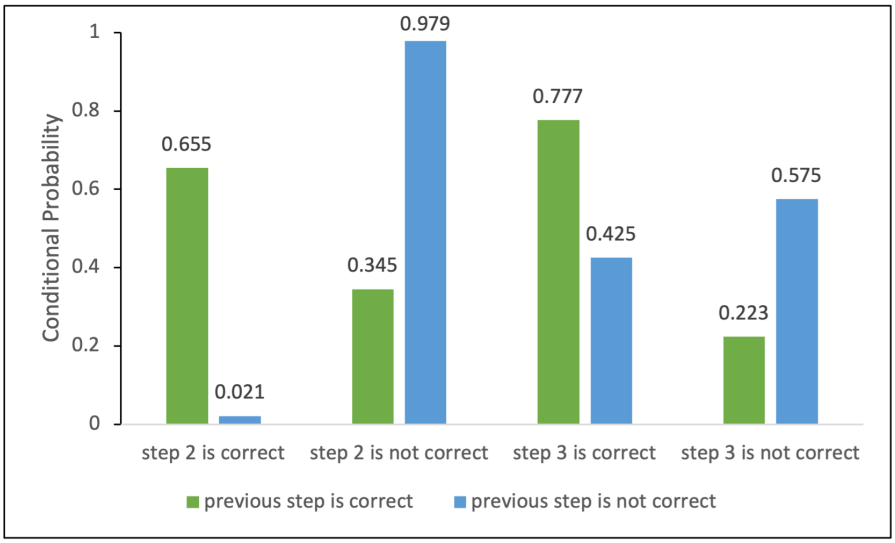

Figure 4.2: Conditional probabilities of prediction behaviors on PQ 3-hop dataset

One explanation is that more hops in reasoning trajectory allow higher variety in potential paths and a larger KG subgraph around the topic entity to be searched, even the agent chooses relations different from the reference ones in step 1 and step 2, it may still figure out the correct relation based on the question.

Another possible reason may be related to the characteristics of the PQ 3-hop, that

the questions are generated with crafted templates. Many queries require meaningless multi-hop, take a question string from PQ dataset as an example, "*what is the dad of daughter of prince christian victor of schleswig-holstein's dad?*", this question can be addressed with a relational chain as "*parent#children#parent*", or any other similar paths involving circular structure.

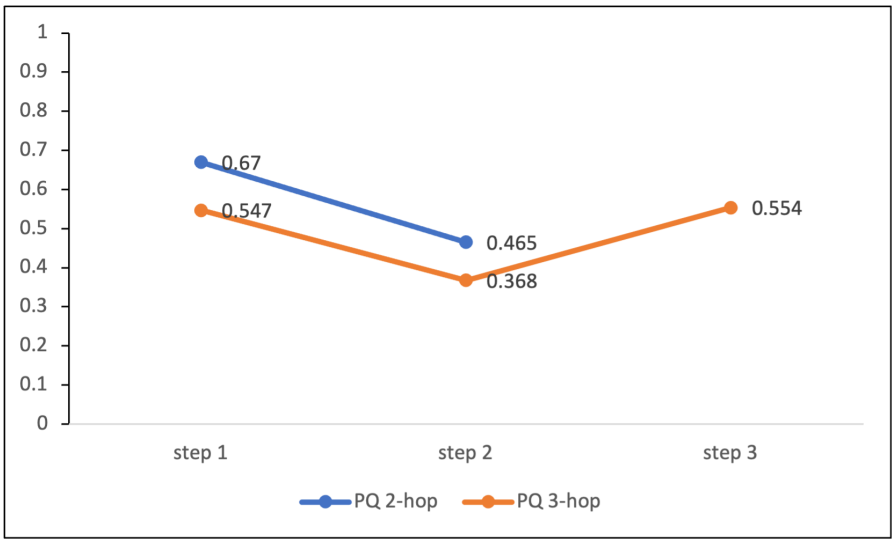

Figure 4.3: The correct ratio in relation prediction on PQ dataset

4.5 Ablation Study

In the RL policy model above, there are two modules designed for investigating the proposed research questions: the attention mechanism and the potential-based reward shaping function. To evaluate the effects of them on model's performance, an ablation study is conducted by disabling the attention module and the reward shaping function respectively.

Model	PQ 2-hop	PQ 3-hop
This Model	55.82	58.95
w/o Attention (Attn)	54.63	58.67
w/o Reward Shaping (RS)	55.68	57.91
w/o Attn and RS	52.97	56.38

Table 4.4: Experimental results on PathQuestion (% hits@1)

RQ: Does history-aware question representation matter in sequential decision making?

As described in the section 3.2.2, the attention layer updates weights of question words in light of action history, and allows the policy network to generate a weighted representation for decision making. The weights themselves are initialized with a uniform distribution. Now the weights updating before each step are neglected, and the hits@1 scores are shown in Table 4.4 and Table 4.5, with the model title "w/o Attention".

Compared with the model setting "w/o Attention and Reward Shaping", employing attention mechanism augments the hits@1 scores by 1 to 2 points generally, which proves that this attention module is beneficial to question representation updating and sequential decision making in resolving the KGQA tasks.

Model	MetaQA 1-hop	MetaQA 2-hop	MetaQA 3-hop
This Model	95.61	48.54	16.71
w/o Attention (Attn)	95.52	46.22	16.33
w/o Reward Shaping (RS)	95.51	47.49	16.12
w/o Attn and RS	93.92	44.45	15.39

Table 4.5: Experimental results on MetaQA (% hits@1)

RQ: Does potential-based reward shaping mechanism affect agent's performance?

The agent receives intermediate shaping rewards generated during state transitions instead of a terminal reward after an episode. To make clear of the effect of potential-based reward shaping on the reinforcement learning process, the shaping rewards are replaced with binary rewards at the end of reasoning paths. The empirical results can be found in Table 4.4 and 4.5 with a setting "w/o Reward Shaping".

Likewise, absorbing potential-based shaping rewards improves the prediction accuracy in two benchmarks. Furthermore, the reward shaping and the attention mechanism are of about the same effectiveness in enhancing the model's performance.

To sum up, the framework presented in section 3 is implemented on two benchmark datasets which are PathQuestions and MetaQA, and the experimental results are listed together with other state-of-the-art architectures previously. The proposed model is analyzed from external and internal aspects, i.e., the final hits@1 scores are compared with the above SOTA frameworks, and the statistics of intermediate actions are visualized and analyzed based on reference gold paths. Based on observations that question mentions tend to have same appearances as their topic entities in general, a passive entity linker is enough to complete the EL subtask in this case. In addition, a series of ablation studies with respects to two research questions were designed and performed, which prove that both two mechanisms have positive impacts on the model's overall performance.

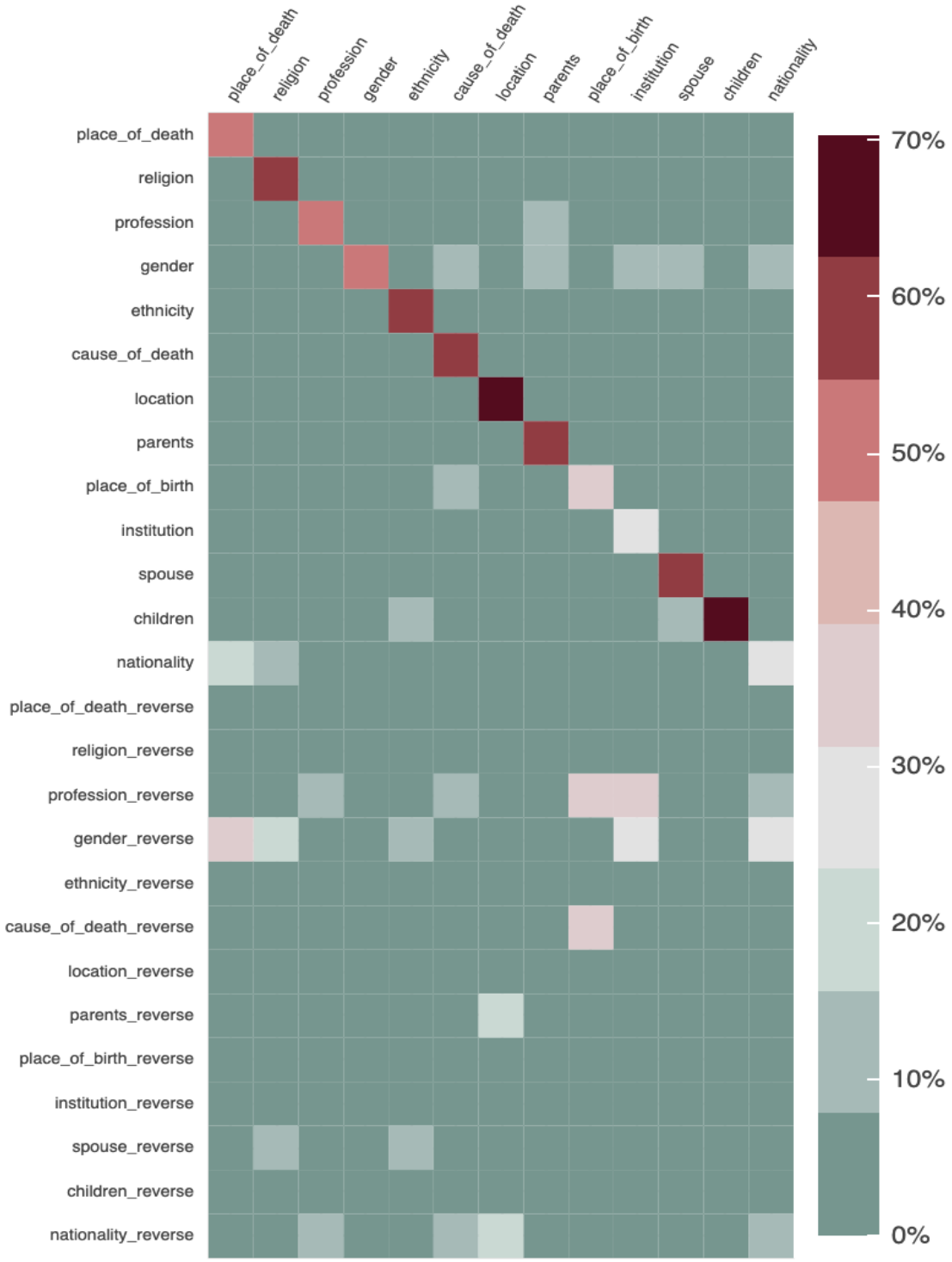

Figure 4.4: The mapping ratios (%) between groundtruth relations (horizontal) and predicted relations (vertical) on PQ 2-hop

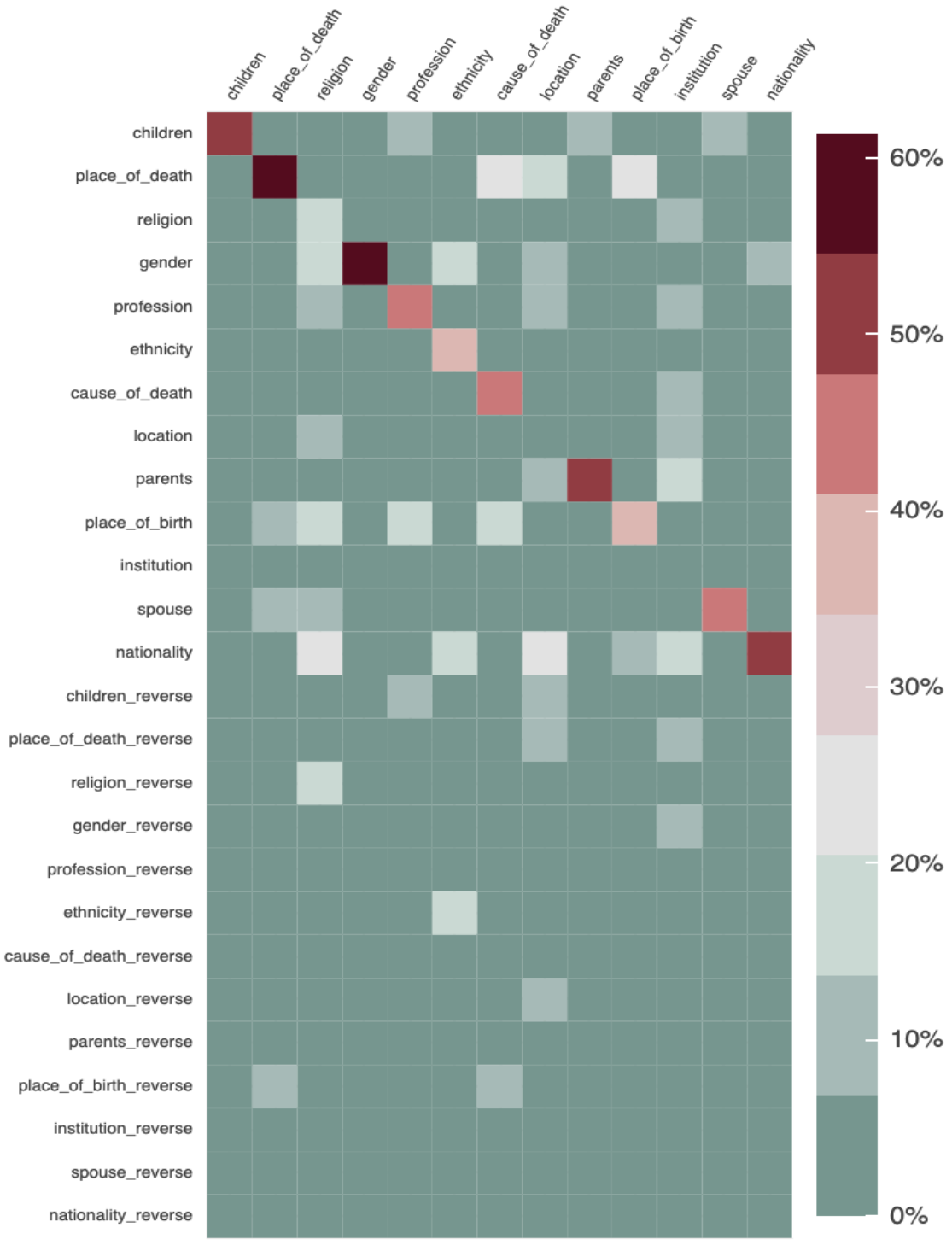

Figure 4.5: The mapping ratios (%) between groundtruth relations (horizontal) and predicted relations (vertical) on PQ 3-hop

5

Limitation Analysis

This section provides a discussion about limitations of the proposed framework based on experimental results.

First, studies above have investigated the effects of the history attention module and potential-based reward shaping on the policy network and proved their contributions on the enhancement of model's performance. One observation from results is that the overall hits@1 performances fall behind the state-of-the-art frameworks by great margins especially on 2-hop and 3-hop QA tasks.

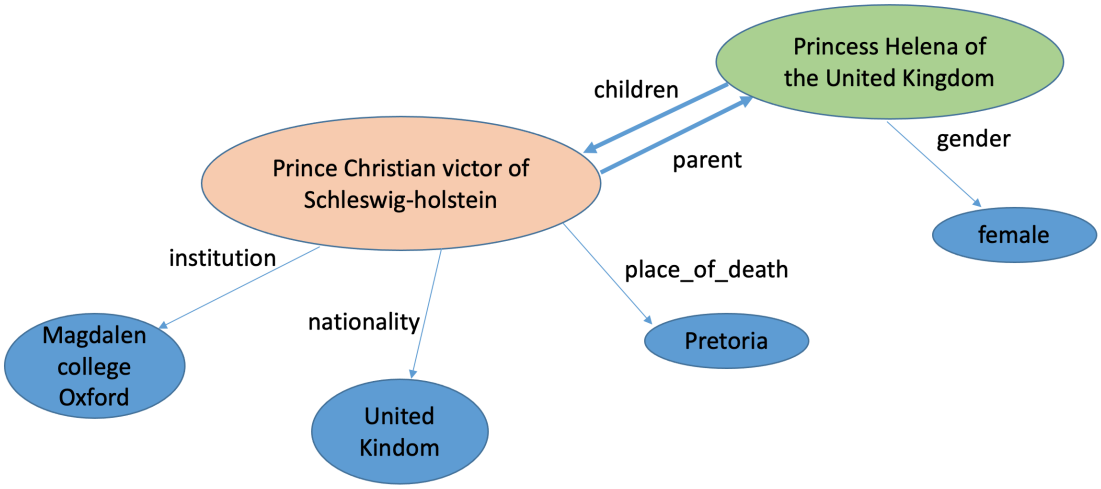

Figure 5.1: An example subgraph for a question from PQ 3-hop: **what is the dad of daughter of prince christian victor of schleswig-holstein's dad?** The orange area denotes the topic entity; the green area represents the corresponding answer entity regarding the question; the bold lines indicate the relations involved in the reference path.

Though the proposed model shows a comparable accuracy over MetaQA 1-hop subset, it does not handle multi-hop questions well, since the accuracy scores drop off drastically on 2-hop and 3-hop subsets compared with that on 1-hop subset. After reviewing the whole model architecture and used datasets, several factors are identified as potentional

reasons why the performance does not live up to the expectations. In the following, the possible influence factors are discussed from two perspectives.

Data Quality

The first argument is that the quality of raw QA datasets can have a deterministic impact on the model's training and testing results, and source data's quality is considered as an external factor.

Though all questions in the QA datasets can be addressed by chains of relations from KGs, the queries are automatically generated with hand-crafted templates, which can result in ambiguities and mistakes in expression. An instance is illustrate in Figure 5.1, where a subgraph from PQ 3-hop knowlegde graph is shown. According to the question, it should be resolved by a sequence of relations as "parent#children#parent", which has a circular path in it. In other words, such circular patterns may mislead the agent in path reasoning over KGs.

Another problem is that the provided answer entities do not always satisfy the conditions in the questions. In the above example, the answer entity has a "gender" relation leading to "female", while the question asks for a "dad" person. Even if a relation type does not make discrimination between entity nodes, e.g., the relation "parent" can connect with either "father" or "mother" entities, the agent has to make a decision based on the query no matter how the predicate "parent" expressed in the question since no further information is available. Hence, it is not rare to observe a mismatch between the question and the predicted relation trajectory by an agent.

Model Competency

The second thought is that the baseline model (without attention and reward shaping) is not good enough. Improvements are observed after applying two mechanisms on the baseline model, which proves their effectiveness to some extend, but the baseline itself does not provide a strong basis.

With the baseline, an input question is tokenized and encoded by a question encoder, and words are weighted by uniform scores in decision making processes, meaning that the agent cannot distinguish between predicates and assign different priorities. Even with the attention mechanism, it can happen that the attentions on different predicates in one question do not vary much at the first step, and a wrong first step tends to cause a wrong second step with a great probability as illustrated in Figure 4.1 and Figure 4.2. In a word, when multiple predicates occur in the same question, **the proposed method is weak in making right first steps in multi-hop tasks**, and this assumption is verified by observations in comparing the predicted paths and the reference paths.

6

Conclusion

This work presents a policy-based reinforcement learning framework in addressing the KGQA problem. Given a question, the RL agent conducts a Markov decision process, which generates a reasoning path leading to a target entity. The model is trained in an end-to-end fashion, that takes natural language questions as inputs and yields answer nodes from the knowledge graph as outputs.

The work evaluates the proposed model on two benchmarks in the experimental studies. The proposed research questions are thoroughly investigated and answered with empirical results. To associate history information with semantic context of questions, a history-aware attention module is implemented to learn to which parts of questions the agent should pay more attention. Furthermore, this work introduces shaping rewards that are proportional to the differences of state potentials, which provides more supervision to RL agent in sequential decison making processes.

The empirical results show that the proposed model has acceptable performances in handling multi-hop questions from two benchmarks, and overall performances benefit from the employment of the attention module and reward shaping mechanism.

Future Perspective

Through an analysis on the model and experimental results, the writer has identified two problems to be further studied. (i) How to assign the attention weights before the initial step? The history-aware attention layer in this model functions after taking encoded history into consideration. However, before history being updated, the agent perceives the question words equally. (ii) How can we leverage beam search in path reasoning? To retrieve a path that most matches with a question, it might be beneficial to keep top-N candidates during extending paths. This strategy compares semantic similarities between a question and a chain of relations instead of a single relation. The two questions have not been investigated in this work, but the author is rather interested in studying the two problems in the future.

References

[Bao et al., 2016] Bao, J., Duan, N., Yan, Z., Zhou, M., and Zhao, T. (2016). Constraint-based question answering with knowledge graph. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2503–2514.

[Bast and Haussmann, 2015] Bast, H. and Haussmann, E. (2015). More accurate question answering on freebase. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1431–1440.

[Bollacker et al., 2008] Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. (2008). Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

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

[Bordes et al., 2013] Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.

[Chen et al., 2019] Chen, Y., Wu, L., and Zaki, M. J. (2019). Bidirectional attentive memory networks for question answering over knowledge bases. *arXiv preprint arXiv:1903.02188*.

[Das et al., 2017] Das, R., Dhuliawala, S., Zaheer, M., Vilnis, L., Durugkar, I., Krishnamurthy, A., Smola, A., and McCallum, A. (2017). Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *arXiv preprint arXiv:1711.05851*.

[Dong and Lapata, 2016] Dong, L. and Lapata, M. (2016). Language to logical form with neural attention. *arXiv preprint arXiv:1601.01280*.

[Fensel et al., 2020] Fensel, D., Simsek, U., and Angele, K. (2020). *Knowledge Graphs: Methodology, Tools and Selected Use Cases*. Springer.

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

[Ganea and Hofmann, 2017] Ganea, O.-E. and Hofmann, T. (2017). Deep joint entity disambiguation with local neural attention. *arXiv preprint arXiv:1704.04920*.

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

[Hildebrandt et al., 2020a] Hildebrandt, M., Li, H., Koner, R., Tresp, V., and Günnemann, S. (2020a). Scene graph reasoning for visual question answering. *arXiv preprint arXiv:2007.01072*.

[Hildebrandt et al., 2020b] Hildebrandt, M., Serna, J. A. Q., Ma, Y., Ringsquandl, M., Joblin, M., and Tresp, V. (2020b). Reasoning on knowledge graphs with debate dynamics. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4123–4131.

[Jin et al., 2021] Jin, W., Yu, H., Tao, X., and Yin, R. (2021). Improving embedded knowledge graph multi-hop question answering by introducing relational chain reasoning. *arXiv preprint arXiv:2110.12679*.

[Kaiser et al., 2021] Kaiser, M., Roy, R. S., and Weikum, G. (2021). Reinforcement learning from reformulations in conversational question answering over knowledge graphs. *arXiv preprint arXiv:2105.04850*.

[Kolitsas et al., 2018] Kolitsas, N., Ganea, O.-E., and Hofmann, T. (2018). End-to-end neural entity linking. *arXiv preprint arXiv:1808.07699*.

[Li et al., 2021] Li, S., Wang, H., Pan, R., and Mao, M. (2021). Memorypath: A deep reinforcement learning framework for incorporating memory component into knowledge graph reasoning. *Neurocomputing*, 419:273–286.

[Liang et al., 2016] Liang, C., Berant, J., Le, Q., Forbus, K. D., and Lao, N. (2016). Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. *arXiv preprint arXiv:1611.00020*.

[Lin et al., 2018] Lin, X. V., Socher, R., and Xiong, C. (2018). Multi-hop knowledge graph reasoning with reward shaping. *arXiv preprint arXiv:1808.10568*.

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

[Mohammed et al., 2017] Mohammed, S., Shi, P., and Lin, J. (2017). Strong baselines for simple question answering over knowledge graphs with and without neural networks. *arXiv preprint arXiv:1712.01969*.

[Ng et al., 1999] Ng, A. Y., Harada, D., and Russell, S. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pages 278–287.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

[Qiu et al., 2020] Qiu, Y., Wang, Y., Jin, X., and Zhang, K. (2020). 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*, pages 474–482.

[Reddy et al., 2014] Reddy, S., Lapata, M., and Steedman, M. (2014). Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics*, 2:377–392.

[Sevgili et al., 2020] Sevgili, O., Shelmanov, A., Arkhipov, M., Panchenko, A., and Biemann, C. (2020). Neural entity linking: A survey of models based on deep learning. *arXiv preprint arXiv:2006.00575*.

[Sukhbaatar et al., 2015] Sukhbaatar, S., Szlam, A., Weston, J., and Fergus, R. (2015). End-to-end memory networks. *arXiv preprint arXiv:1503.08895*.

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

[Sutton and Barto, 2018] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

[Wang et al., 2019] Wang, H., Li, S., Pan, R., and Mao, M. (2019). Incorporating graph attention mechanism into knowledge graph reasoning based on deep reinforcement learning. 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 2623–2631.

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

[Wong and Mooney, 2007] Wong, Y. W. and Mooney, R. (2007). Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 960–967.

[Xiong et al., 2017] Xiong, W., Hoang, T., and Wang, W. Y. (2017). Deeppath: A reinforcement learning method for knowledge graph reasoning. *arXiv preprint arXiv:1707.06690*.

[Yao and Van Durme, 2014] Yao, X. and Van Durme, B. (2014). 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)*, pages 956–966.

[Yih et al., 2015] Yih, S. W.-t., Chang, M.-W., He, X., and Gao, J. (2015). Semantic parsing via staged query graph generation: Question answering with knowledge base.

[Yin et al., 2016] Yin, W., Yu, M., Xiang, B., Zhou, B., and Schütze, H. (2016). Simple question answering by attentive convolutional neural network. *arXiv preprint arXiv:1606.03391*.

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

[Zhou et al., 2018] Zhou, M., Huang, M., and Zhu, X. (2018). An interpretable reasoning network for multi-relation question answering. *arXiv preprint arXiv:1801.04726*.

A

Appendix

The complete code, datasets, results of the experiments, and implementation descriptions are available at *https://gitlab.i .uzh.ch/ddis/Students/Theses/2021-fan-feng*.

List of Figures

List of Tables