UC Santa Barbara

UC Santa Barbara Electronic Theses and Dissertations

Title

Neural Question Answering Models with Broader Knowledge Scope and Deeper Reasoning Power

Permalink

https://escholarship.org/uc/item/816853qx

Author

Xiong, Wenhan

Publication Date

2021

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

University of California
Santa Barbara

Neural Question Answering Models with Broader Knowledge Scope and Deeper Reasoning Power

A dissertation submitted in partial satisfaction
of the requirements for the degree

Doctor of Philosophy
in
Computer Science

by

Wenhan Xiong

Committee in charge:

Professor William Wang, Chair
Professor Xifeng Yan
Professor Yu-Xiang Wang

March 2021

The Dissertation of Wenhan Xiong is approved.

Professor Xifeng Yan

Professor Yu-Xiang Wang

Professor William Wang, Committee Chair

March 2021

Neural Question Answering Models

with Broader Knowledge Scope and Deeper Reasoning Power

Copyright © 2021

by

Wenhan Xiong

To my grandma

Acknowledgements

The past five years at Santa Barbara has been a fascinating experience in my life. Studying and doing research alongside the beautiful beaches is definitely a luxurious privilege. But it is the people here at UCSB that make me feel even more fortunate.

Before I started my PhD, I almost had zero experience of doing any kind of AI research, not to mention NLP. In retrospect, I feel very fortunate about the opportunity that put me into this existing area. I am extremely grateful to my advisor William Wang, who took chance on me despite my totally irrelevant academic background and helped me publish my first paper in just three months. For a junior student still trying to figure out what exactly is research, that paper and the later oral talk at the conference greatly boosted my confidence. Throughout my whole PhD, William's open-mindedness and encouragement are priceless to me, especially during those bumpy times. Without his help, I would never have the persistence to finish my PhD. I am also indebted to Prof. Xifeng Yan for all the advice he offered me at the very beginning of my PhD, although I believe it was not until later that I began to get the message. He was also the person who handed the first QA paper to me and inspired my initial interests in this field. I would also like to thank Prof. Yu-Xiang Wang for taking time to be on my thesis committee and providing insights from a theoretical perspective.

Besides my mentors on campus, I am grateful to my wonderful mentors during internships – Mo Yu, Xiaoxiao Guo, Shiyu Chang, Caiming Xiong, Jingfei Du, Ves Stoyanov, Scott Yih, Barlas Oğuz, Srini Iyer, Sebastian Riedel and Douwe Kiela for their invaluable inspiration, support and guidance. I would like to thank my labmates and long-time collaborators: Hong Wang, Wenhu Chen, Xin Wang, Jiawei Wu; and other awesome coauthors: Xiang Lorraine Li, Patrick Lewis, Yashar Mahdad and Murray Campbell, for the fruitful discussions.

Finally, I would like to thank my parents for their love. It was my mother's insistence on giving me a better education at an earlier age that made this whole journey possible. I am grateful to my grandparents, who gave me a carefree childhood. Lastly, I feel extremely fortunate to have Xiaolu by my site over the past eight years. While sometimes I could be really depressed and pessimistic about various things in life and study, she seems to be a born optimist. She always has the ability to find the spark among darkness. It was her accompany and love that gave me the most power to overcome everything over the years.

Abstract

Neural Question Answering Models

with Broader Knowledge Scope and Deeper Reasoning Power

by

Wenhan Xiong

Natural language has long been the most prominent tool for humans to disseminate, learn and create knowledge. However, in the era where new information is generated at an unprecedentedly rate and people's craving of knowledge becomes broader and deeper, efficiently extracting the desired knowledge from the vast amount of language become a significant challenge. Even with the aid of modern search engines which sometimes directly return a text snippet along with the ranked list of pages. The accuracy of extracted knowledge is still insufficient such that the users often need to manually inspect each of the retrieved pages. This is especially the case when the queries becomes more complex and less common.

In this dissertation, we investigate the problem of knowledge extraction centering around a simple and generic task formulation: we aim to build a system that takes natural language questions as input, processes the underlying knowledge source (usually text corpus or structured knowledge source) and finally returns a short piece of text that adequately answers the questions. In a nutshell, the goal of the proposed approaches in this dissertation is to enable AI systems to accurately answer broader and harder questions. We begin by studying the traditional structured QA system which uses a structured knowledge base (KB) as the underlying knowledge source. Specifically, we propose reasoning methods to automatically populate the missing knowledge in a KB, and a hybrid neural model that combines both KB and text to answer questions. Next,

we utilize strong and large pretrained models to build QA systems that directly answer questions from text corpora. We introduce a knowledge-enhanced pretraining strategy which explicitly injects more entity-centric knowledge into pretrained models. Finally, we present a multi-hop QA model that could efficiently navigate over the large text corpus (over millions of documents) and reason over multiple text evidence to derive the answer. Altogether, these techniques allow users to ask questions from broader domains and with increased complexity.

Contents

Abstract **vii**

1 Introduction **1**
1.1 Motivation 1
1.2 Background 3
1.3 Challenges 6
1.4 Overview 7

Part I Improve the Knowledge Scope of KBQA Systems **9**

2 Automatic KB Completion via Multi-Hop Reasoning **10**
2.1 Introduction 10
2.2 Related Work 12
2.3 Learning to Reason with Deep Reinforcement Learning 13
2.4 Experiments 21

3 Few-Shot Learning for KB Completion **27**
3.1 Introduction 27
3.2 Related Work 28
3.3 Few-Shot Relational Learning 30
3.4 Experiments 37
3.5 Summary on KB Completion 44

4 Augment KBQA with Knowledge-Aware Text Reader **45**
4.1 Introduction 45
4.2 Task Definition 46
4.3 QA with Hybrid Knowledge Source 47
4.4 Experiments 51

Part II Efficient and Deep QA from Text **54**

5 Learning Dense Corpus Representations for Efficient Retrieval **55**
5.1 Introduction 55
5.2 Related Work 57
5.3 Progressively Pretrained Dense Corpus Representation 58
5.4 Experiments 65

6 Improving Machine Reading via Knowledge-Aware Pretraining **73**
6.1 Introduction 73
6.2 Related Work 75
6.3 Entity Replacement Training 76
6.4 Experiments 79

7 Multi-Hop Reasoning for Complex Textual QA **88**
7.1 Introduction 88
7.2 Related Work 90
7.3 Recursive Dense Retrieval 92
7.4 Experiments 94

8 Conclusions **109**

Bibliography **112**

Chapter 1

Introduction

1.1 Motivation

It is believed that the Homo sapiens is distinguished from other species by the ability of using natural language [1]. Since the very early days of AI, building machines that understand human language has been one of the long-standing research themes. In 1950, Alan Turing proposed the Turing Test, largely based on natural language: a human and a computer communicate in natural language; and the human evaluator would judge the emergence of intelligence according to whether the computer's behaviour is indistinguishable from other human participants. Over the past several decades, the techniques to build such machines have evolved from the earliest attempts of rule-based approaches, e.g., the SHRDLU system developed in the 1960s allows users to give instructions in a highly restricted environment with limited vocabularies, to modern statistical methods, e.g., the IBM alignment models for statistical machine translation. More recently, this trend have been pushed to an extreme with the breakthrough of deep neural models, especially these powerful pretrained language models which can easily scale to over billions of parameters learned from large-scale self-supervised training.

While there is no unique and concrete definition of *language understanding* in the NLP community, the notion of language understanding, as well as the various NLP tasks, could be broken down into different levels. At lower levels, researchers have built tools that could assign labels (i.e., part-of-speech tags) to each of the tokens, indicating whether the token is part of a *noun*, *verb*, etc. On top of the token-level labelling tasks, there are parsing tasks which aim to understand the syntactic/grammatical dependencies between tokens and phrases. At higher levels, researchers are interested in the semantics (meaning) of language, such the semantic roles (e.g., labels that indicate *who* did *what* to *whom* at *where*?). While the tasks across different levels have their own research values, the problems we studied at higher levels are often closer to practical applications, as language is essentially a tool for humans to communicate. In most cases, humans will not try to decipher the underlying syntactic or grammar structures of the words. Instead, we are directly extracting the important message without building an explicit syntactic model in our minds. Additionally, well-designed high-level tasks could actually embody the skills required for lower level tasks. For example, answering questions typically involves the skills of named entity recognition and coreference resolution. Motivated by these observations, this dissertation will be focusing on solving high-level tasks that are close to the original purpose of natural language, i.e., exchanging information and knowledge.

Among all kinds of high-level semantic-related tasks, the task of QA is simple yet general. Its simple formulation (e.g., extracting arbitrary spans or entities) allows us to define straightforward and reliable evaluation protocols, and the annotation of the ground-truth labels could be efficiently scaled. The QA task is also general enough upon which many real-world applications (such as intelligent search engines, virtual assistant) could be built. While many tasks, such as sentiment analysis and fact verification, take a simpler formulation of multi-category classification and capture certain aspects of the language, standard QA formulation can involves a wider-range and finer-grained language

understanding. For instance, while a fact verification systems only predict an *False* label to a claim, e.g., `Nikolaj Coster-Waldau worked with the Fox Broadcasting Company.`, a QA system can tell us how to correct the claim, by asking a question about the salient entity in the claim, e.g., `Which company did Nikolaj Coster-Waldau work with?`. Indeed, semantic-level tasks can usually be easily convert to a QA format. For instance, to verify a factual claim, we could trivially convert it into a yes/no question. On the other hand, to answer an arbitrary question, since the candidate pool is not known beforehand, it is typically impossible to solve the QA task under a simple classification formulation. Overall, we believe question answering is a task of great values for both research and practical applications.

1.2 Background

Before diving into the proposed methods, we first introduce the existing techniques on top of which the proposed approaches are built. We briefly review question answering systems based on knowledge bases and the recent proposed machine reading models that allow accurate answer extraction from text.

1.2.1 Knowledge Base Question Answering

The main motivation of constructing knowledge bases is to resolve the ambiguity in natural language and grounding the semantics into a predefined set of symbols (relations and entities). For instance, in documents about Tom Hanks, the established actor could be mentioned as "Thomas Jeffrey Hanks", "Hanks" or simply "He" in different contexts. This is obviously an inefficient way to organize knowledge due to the redundancies in text data. More importantly, the free-form expressions of the same entity makes it challenging to direct retrieve knowledge from text in the original unstructured format, as

the mentions in the query might not match the mentions in the answer context. Briefly, popular knowledge bases used for QA, such as Freebase [2], DBpedia [3] and Wikidata [4] are organized as a huge graph, where the nodes are typically real-world entities and the directional edges represent the relations between the entity pairs. Besides these binary relations, there are other types of edges that store some properties of the entities, such as alias, types or wikipedia links. To answer questions, the system needs to ground the free-form mentions of entities and relations into a KB subgraph. The answers are usually a single or a set of the entities connected by the relations detected in the query.

The mainstream technology to build KBQA systems is known as semantic parsing [5, 6, 7]. The parser aims to convert the natural language question (also referred as the *utterance*) into an unambiguous logical form such as λ-calculus or λ-DCS [8]. Consider the question *"How many senators are born in Santa Barbara?"*, the desired semantic parse in λ-calculus is:

$$\texttt{count}(\lambda x.\texttt{Profession}(x, \texttt{senator}) \wedge \texttt{PlaceofBirth}(x, \texttt{SantaBarbara})$$

These formal languages are finally mapped to graph query languages (e.g., SPARQL) which can be executed against a graph database that holds the KB data, and retrieve the set of entities that satisfy the logical conditions. One of the main challenges of this work was the lack of human annotated logical forms for training, as the labeling process requires expertise in the logical language. To learn such parsers directly from question-answer pairs, existing work [9, 10] usually use weakly supervised logical forms (whose execution match the ground-truth answers) for training. More recently, reinforcement learning has also been adopted for the learning problem [11].

While semantic-parsing based approaches have achieved superior performance on KBQA, these systems are typically highly engineered and require various hand-designed

features. Another paradigm of building KBQA systems is based on end-to-end learned embedding representations [12]. The symbols in KBs and the word tokens are represented by low-dimensional continuous vectors, which are used in a matching function between the query the answer candidates. At inference time, the *topic entity* in the question is recognized by an entity linking module. The entities in the *topic entity*'s local subgraph are considered as answer candidates. To make the embedding model expressive enough, the local structures, such as neighbors of candidate entities and the relational path from the *topic entity*, are encoded in the answer-entity representations. More recently, graph neural networks have also been used to learn such continuous representations [13]

1.2.2 Neural Reading Comprehension

The recent success of recent neural reading comprehension models is largely due to the availability of large-scale datasets, which allow the training of powerful deep learning models. The first large-scale reading comprehension task focuses on cloze-style questions [14]. This design allows the efficient construction of large amount of data, by using the bullet points of news articles to automatically to create questions. However, the automatic data collection process is noisy and also fail to capture many phenomena in natural language questions [15]. SQuAD [16] is the first large-scale crowdsourced dataset that spawned the breakthroughs of many recent reading comprehension models. The questions are human written upon a given Wikipedia passage and the answers are continuous spans. Compared to multiple-choice answers, the format of answer spans allows the models to be used in more practical scenarios where the answer candidates are not known. This design is also more controllable compared to free-form answer annotations, which can be difficult evaluate against when there exist multiple correct answers.

Early breakthroughs on reading comprehension are based on complex attention lay-

ers [17] which learn fine-grained matching between the question and passage tokens. The core ingredient of these models is the bidirectional token-level attention between the question and passage. On top of the final passage representations produced by these attention layers, two linear layers are applied to predict the answer start and end positions. These models push the answer accuracy to 60-70% percent from the initial 40% accuracy from baseline systems. More recently, human-level performance has been achieved by large pretrained language models, such as BERT [18]. These models rely on a solely attention based encoder, called Transformer [19], that can effectively scales to large-scale datasets and very deep layers. These models are pretrained on large corpus in a self-supervised fashion: the models are trained to predict masked tokens from the bidirectional context. Thus, no human annotation is required and corpora with billions of tokens can be used. It turned out that once properly scaled, such pretrained models could bring significant improvements across all kinds of NLP tasks. In the context of this dissertation, the nearly perfect performance on the reading comprehension task has made these models the indispensable components to build text-based QA systems. More importantly, as pretrained models usually do not require carefully designed architectures to work well, they have largely democratized the pipelines to build state-of-the-art QA applications.

1.3 Challenges

The structured knowledge source make it easier for machines to query, as the system does not need to handle the large amount of text documents. Since related facts are usually organized together in a local subgraph, it is also easier to answer questions that involves multiple relations as related facts. Besides, the database query language allows the system to trivially answer questions that requires aggregation, such as `argmax`, `count` , etc. The most prominent limitation of KBQA systems is the knowledge scope of the

underlying knowledge source. Compared to the abundant text data on the Web, KBs only cover a limited set of relations and cannot be used to find answers beyond real-world entities, e.g., why questions. To ensure precision, the construction of KBs relies on lots of human efforts, which make them hard to capture up-to-date knowledge. Automatic KB completion methods need to be faithful and explainable in order to maintain the KBs' prevision. However, learning fully interpretable KB completion methods requires weakly supervised training, which typically results in inferior performance compared to black-box embedding-based methods. Moreover, real-world scenarios usually demands few-shot learning, because most of the KB relations are long-tail.

On the contrary, the main difficulties of lies in the processing of the unstructured text corpora, which can include billions of documents. First of all, a retrieval module is required to locate the relevant documents. The retrieval system should be general enough to model the high-level semantics expressed in various language patterns. It also needs to be expressive enough to capture the nuance in language that causes dramatic semantic shift. Most importantly, the retrieval needs to highly efficient that handles the large amount of documents. Second, compared to KBQA, it is nontrivial to answer questions that require the aggregation of multiple facts. In the most general and practical setup, the text corpus is simply a huge list of documents, and the inter-document relations do not exist. Thus, the required evidence might disperse in distant documents. The QA system needs to efficiently connect two relevant documents with respect to the questions and model the interactions to accurate predict the answer.

1.4 Overview

In the following chapters, we present a set of individual methods, including new model architectures, learning algorithms, efficient inference procedures, etc., that tackle

the aforementioned challenges in both QA paradigms. We begin by investigating the structured QA approach: a large scale of knowledge base is first built from text and then neural models are built to extract answers from the KB. Here we aim at improving the scope of the structured knowledge source and enabling the system to answer questions on more topics (Chapter 2, 3, 4). Afterwards, we study techniques that leverage the representation power of large pretrained models to directly extract answers from large-scale text corpus. The abundant text data on the Web brings better knowledge coverage than the KB approach. However, the ambiguity of natural language and the isolation of each knowledge source makes it challenging to locate the answer context. We propose effective techniques that improve the document retrieval and answer extraction modules (Chapter 5, 6). Moreover, we introduce a strong multi-hop retrieval system could retrieval relevant documents in a recursive fashion, allowing the QA system to efficiently reason over multiple evidences to answer more complex questions (Chapter 7).

Part I

Improve the Knowledge Scope of KBQA Systems

Chapter 2

Automatic KB Completion via Multi-Hop Reasoning

2.1 Introduction

Due to the costly process of building and maintaining structured knowledge bases, it is challenging to ensure a broad and up-to-date knowledge scope. However, with deep learning techniques such as distributional symbol representations, we could augment KBQA systems with automatic KB completion methods that alleviate the knowledge coverage issues . For one direction, we could use *information extraction* (IE) techniques that extract entities and relations from text and then add the extracted triples to the existing KBs. However, training such IR models typically rely on distant supervision and a error propagating pipeline (e.g., named entity recognition, linking and relation extraction). In this dissertation (this and the following chapter), we take an alternative and simpler paradigm which makes use of the KB itself to obtain training supervision. We aim to develop machine learning systems that could infer new knowledge by reasoning over the existing structured triples.

Broadly, deep learning techniques have obtained many state-of-the-art results in various classification and recognition problems [20, 21, 22], mainly due to the representation power of highly-parametrized non-linear neural networks. However, complex reasoning problems often require multiple inter-related decisions, and empowering deep learning models with the ability of learning to reason is still a challenging issue. To handle complex queries where there are no obvious answers, intelligent machines must be able to reason with existing resources, and learn to infer an unknown answer.

More specifically, we situate our study in the context of multi-hop reasoning for knowledge base completion, which is the task of learning explicit inference formulas, given a large KG. For example, if the KG includes the beliefs such as *Neymar* plays for *Barcelona*, and *Barcelona* are in the *La Liga* league, then machines should be able to learn the following formula: $playerPlaysForTeam(P,T) \wedge teamPlaysInLeague(T,L) \Rightarrow playerPlaysInLeague(P,L)$. In the testing time, by plugging in the learned formulas, the system should be able to automatically infer the missing link between a pair of entities. This kind of reasoning machine can serve as an essential components of complex QA systems.

Recently, the Path-Ranking Algorithm (PRA) [23, 24] emerges as a promising method for learning inference paths in large KGs. PRA uses a random-walk with restarts based inference mechanism to perform multiple bounded depth-first search processes to find relational paths. Coupled with elastic-net based learning, PRA then picks more plausible paths using supervised learning. However, PRA operates in a fully discrete space, which makes it difficult to evaluate and compare similar entities and relations in a KG.

In this chapter, we propose a novel approach for controllable multi-hop reasoning: we frame the path learning process as reinforcement learning (RL). In contrast to PRA, we use translation-based knowledge based embedding method [25] to encode the continuous state of our RL agent, which reasons in the vector space environment of the knowledge

graph. The agent takes incremental steps by sampling a relation to extend its path. To better guide the RL agent for learning relational paths, we use policy gradient training [26] with a novel reward function that jointly encourages accuracy, diversity, and efficiency. Empirically, on the task of knowledge base completion, we show that our method outperforms PRA and embedding based methods on Freebase and the Never-Ending Language Learning [27] knowledge base.

2.2 Related Work

The Path-Ranking Algorithm (PRA) method [28] is a primary path-finding approach that uses random walk with restart strategies for multi-hop reasoning. Gardner et al. [29, 30] propose a modification to PRA that computes feature similarity in the vector space. Wang and Cohen [31] introduce a recursive random walk approach for integrating the background KG and text—the method performs structure learning of logic programs and information extraction from text at the same time. A potential bottleneck for random walk inference is that supernodes connecting to large amount of formulas will create huge fan-out areas that significantly slow down the inference and affect the accuracy.

Toutanova et al. [32] provide a convolutional neural network solution to multi-hop reasoning. They build a CNN model based on lexicalized dependency paths, which suffers from the error propagation issue due to parse errors. Guu et al. [33] uses KG embeddings to answer path queries. Zeng et al. [34] described a CNN model for relational extraction, but it does not explicitly model the relational paths. Neelakantan et al. [35] propose a recurrent neural networks model for modeling relational paths in knowledge base completion (KBC), but it trains too many separate models, and therefore it does not scale. Note that many of the recent KG reasoning methods [35, 36] still rely on first learning the PRA paths, which only operates in a discrete space. Comparing to PRA,

our method reasons in a continuous space, and by incorporating various criteria in the reward function, our reinforcement learning (RL) framework has better control and more flexibility over the path-finding process.

Neural symbolic machine [37] is a more recent work on KG reasoning, which also applies reinforcement learning but has a different flavor from our work. NSM learns to compose programs that can find answers to natural language questions, while our RL model tries to add new facts to knowledge graph (KG) by reasoning on existing KG triples. In order to get answers, NSM learns to generate a sequence of actions that can be combined as a executable program. The action space in NSM is a set of predefined tokens. In our framework, the goal is to find reasoning paths, thus the action space is relation space in the KG. A similar framework [38] has also been applied to visual

2.3 Learning to Reason with Deep Reinforcement Learning

In this section, we describe in detail our RL-based framework for multi-hop relation reasoning. The specific task of relation reasoning is to find reliable predictive paths between entity pairs. We formulate the path finding problem as a sequential decision making problem which can be solved with a RL agent. We first describe the environment and the policy-based RL agent. By interacting with the environment designed around the KG, the agent learns to pick the promising reasoning paths. Then we describe the training procedure of our RL model. After that, we describe an efficient path-constrained search algorithm for relation reasoning with the paths found by the RL agent.

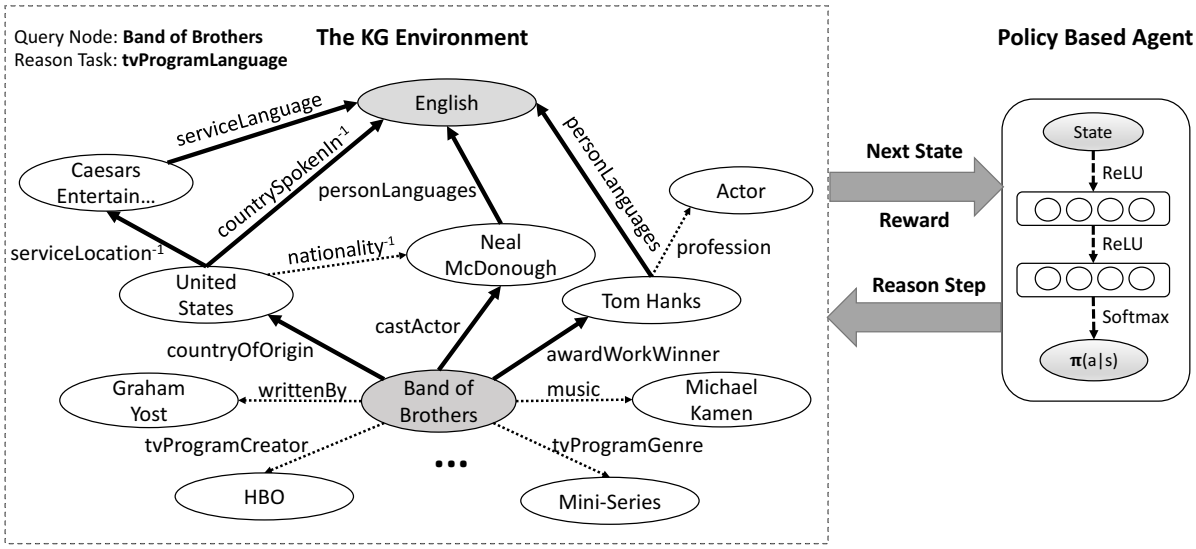


Figure 2.1: Overview of our RL model. **Left:** The KG environment $\mathcal{E}$ modeled by a MDP. The dotted arrows (partially) show the existing relation links in the KG and the bold arrows show the reasoning paths found by the RL agent. $^{-1}$ denotes the inverse of an relation. **Right:** The structure of the policy network agent. At each step, by interacting with the environment, the agent learns to pick a relation link to extend the reasoning paths.

2.3.1 Reinforcement Learning for Relation Reasoning

The RL system consists of two parts (see Figure 2.1). The first part is the external environment $\mathcal{E}$ which specifies the dynamics of the interaction between the agent and the KG. This environment is modeled as a Markov decision process (MDP). A tuple $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} >$ is defined to represent the MDP, where $\mathcal{S}$ is the continuous state space, $\mathcal{A} = \{a_1, a_2, ..., a_n\}$ is the set of all available actions, $\mathcal{P}(S_{t+1} = s^{'}|S_t = s, A_t = a)$ is the transition probability matrix, and $\mathcal{R}(s, a)$ is the reward function of every (s, a) pairs.

The second part of the system, the RL agent, is represented as a policy network $\pi_\theta(s, a) = p(a|s; \theta)$ which maps the state vector to a stochastic policy. The neural network parameters θ are updated using stochastic gradient descent. Compared to Deep Q Network (DQN) [39], policy-based RL methods turn out to be more appropriate for our knowledge graph scenario. One reason is that for the path finding problem in KG, the action space can be very large due to complexity of the relation graph. This can lead to poor convergence properties for DQN. Besides, instead of learning a greedy policy which is common in value-based methods like DQN, the policy network is able to learn a stochastic policy which prevent the agent from getting stuck at an intermediate state. Before we describe the structure of our policy network, we first describe the components (actions, states, rewards) of the RL environment.

Actions Given the entity pairs (e_s, e_t) with relation r, we want the agent to find the most informative paths linking these entity pairs. Beginning with the source entity e_s, the agent use the policy network to pick the most promising relation to extend its path at each step until it reaches the target entity e_t. To keep the output dimension of the policy network consistent, the action space is defined as all the relations in the KG.

States The entities and relations in a KG are naturally discrete atomic symbols. Since existing practical KGs like Freebase [2] and NELL [40] often have huge amounts of triples. It is impossible to directly model all the symbolic atoms in states. To capture the semantic information of these symbols, we use translation-based embeddings such as TransE [25] and TransH [41] to represent the entities and relations. These embeddings map all the symbols to a low-dimensional vector space. In our framework, each state captures the agent's position in the KG. After taking an action, the agent will move from one entity to another. These two are linked by the action (relation) just taken by the agent. The state vector at step t is given as follows:

$$\mathbf{s}_t = (\mathbf{e}_t, \mathbf{e}_{target} - \mathbf{e}_t) \tag{2.1}$$

where $\mathbf{e}_t$ denotes the embeddings of the current entity node and $\mathbf{e}_{target}$ denotes the embeddings of the target entity. At the initial state, $\mathbf{e}_t = \mathbf{e}_{source}$. We do not incorporate the reasoning relation in the state, because the embedding of the reasoning relation remain constant during path finding, which is not helpful in training. However, we find out that by training the RL agent using a set of positive samples for one particular relation, the agent can successfully discover the relation semantics.

Rewards There are a few factors that contribute to the quality of the paths found by the RL agent. To encourage the agent to find predictive paths, our reward functions include the following scoring criteria:

Global accuracy: For our environment settings, the number of actions that can be taken by the agent can be very large. In other words, there are much more incorrect sequential decisions than the correct ones. The number of these incorrect decision sequences can increase exponentially with the length of the path. In view of this challenge, the first

reward function we add to the RL model is defined as follows:

$$r_{\text{GLOBAL}} = \begin{cases} +1, & \text{if the path reaches } e_{target} \\ -1, & \text{otherwise} \end{cases}$$

the agent is given an offline positive reward +1 if it reaches the target after a sequence of actions.

Path efficiency: For the relation reasoning task, we observe that short paths tend to provide more reliable reasoning evidence than longer paths. Shorter chains of relations can also improve the efficiency of the reasoning by limiting the length of the RL's interactions with the environment. The efficiency reward is defined as follows:

$$r_{\text{EFFICIENCY}} = \frac{1}{length(p)}$$

where path p is defined as a sequence of relations $r_1 \rightarrow r_2 \rightarrow ... \rightarrow r_n$.

Path diversity: We train the agent to find paths using positive samples for each relation. These training sample (e_{source}, e_{target}) have similar state representations in the vector space. The agent tends to find paths with similar syntax and semantics. These paths often contains redundant information since some of them may be correlated. To encourage the agent to find diverse paths, we define a diversity reward function using the cosine similarity between the current path and the existing ones:

$$r_{\text{DIVERSITY}} = -\frac{1}{|F|} \sum_{i=1}^{|F|} cos(\mathbf{p}, \mathbf{p}_i)$$

where $\mathbf{p} = \sum_{i=1}^{n} \mathbf{r}_i$ represents the path embedding for the relation chain $r_1 \rightarrow r_2 \rightarrow ... \rightarrow r_n$.

Policy Network We use a fully-connected neural network to parameterize the policy

function $\pi(s;\theta)$ that maps the state vector $\mathbf{s}$ to a probability distribution over all possible actions. The neural network consists of two hidden layers, each followed by a rectifier nonlinearity layer (ReLU). The output layer is normalized using a softmax function (see Figure 2.1).

2.3.2 Training Pipeline

In practice, one big challenge of KG reasoning is that the relation set can be quite large. For a typical KG, the RL agent is often faced with hundreds (thousands) of possible actions. In other words, the output layer of the policy network often has a large dimension. Due to the complexity of the relation graph and the large action space, if we directly train the RL model by trial and errors, which is typical for RL algorithms, the RL model will show very poor convergence properties. After a long-time training, the agents fails to find any valuable path. To tackle this problem, we start our training with a supervised policy which is inspired by the imitation learning pipeline used by *AlphaGo* [42]. In the Go game, the player is facing nearly 250 possible legal moves at each step. Directly training the agent to pick actions from the original action space can be a difficult task. *AlphaGo* first train a supervised policy network using experts moves. In our case, the supervised policy is trained with a randomized breadth-first search (BFS).

Supervised Policy Learning For each relation, we use a subset of all the positive samples (entity pairs) to learn the supervised policy. For each positive sample (e_{source}, e_{target}), a two-side BFS is conducted to find same correct paths between the entities. For each path p with a sequence of relations $r_1 \rightarrow r_2 \rightarrow ... \rightarrow r_n$, we update the parameters θ to maximize the expected cumulative reward using Monte-Carlo Policy Gradient (REIN-

FORCE) [43]:

$$\begin{aligned} J(\theta) &= \mathbb{E}_{a\sim\pi(a|s;\theta)}(\sum_t R_{s_t,a_t}) \\ &= \sum_t \sum_{a\in\mathcal{A}} \pi(a|s_t;\theta) R_{s_t,a_t} \end{aligned} \tag{2.2}$$

where $J(\theta)$ is the expected total rewards for one episode. For supervised learning, we give a reward of $+1$ for each step of a successful episode. By plugging in the paths found by the BFS, the approximated gradient used to update the policy network is shown below:

$$\begin{aligned} \nabla_\theta J(\theta) &= \sum_t \sum_{a\in\mathcal{A}} \pi(a|s_t;\theta) \nabla_\theta \log \pi(a|s_t;\theta) \\ &\approx \nabla_\theta \sum_t \log \pi(a = r_t|s_t;\theta) \end{aligned} \tag{2.3}$$

where r_t belongs to the path p.

However, the vanilla BFS is a biased search algorithm which prefers short paths. When plugging in these biased paths, it becomes difficult for the agent to find longer paths which may potentially be useful. We want the paths to be controlled only by the defined reward functions. To prevent the biased search, we adopt a simple trick to add some random mechanisms to the BFS. Instead of directly searching the path between e_{source} and e_{target}, we randomly pick a intermediate node e_{inter} and then conduct two BFS between (e_{source}, e_{inter}) and (e_{inter}, e_{target}). The concatenated paths are used to train the agent. The supervised learning saves the agent great efforts learning from failed actions. With the learned experience, we then train the agent to find desirable paths.

Retraining with Rewards To find the reasoning paths controlled by the reward functions, we use reward functions to retrain the supervised policy network. For each relation, the reasoning with one entity pair is treated as one episode. Starting with the source

Algorithm 1 Retraining Procedure with reward functions

Restore parameters θ from supervised policy

for episode $\leftarrow$ 1:N **do**
  Initialize state vector $s_t \leftarrow s_0$
  Initialize episode length $steps \leftarrow 0$
  while $num_steps < max_length$ **do**
    Randomly sample action $a \sim \pi(a|s_t)$
    Observe reward $\mathcal{R}_t$, next state s_{t+1}
    if $\mathcal{R}_t = -1$ **then**
      Save $< s_t, a >$ to $\mathcal{M}_{neg}$
    end if
    if success or $steps = max_length$ **then**
      break
    end if
    Increment num_steps
  end while
  Update θ using $g \propto \nabla_\theta \sum_{\mathcal{M}_{neg}} \log \pi(a = r_t|s_t; \theta)(-1)$
  if success **then**
    $R_{total} \leftarrow \lambda_1 r_{\text{GLOBAL}} + \lambda_2 r_{\text{EFFICIENCY}} + \lambda_3 r_{\text{DIVERSITY}}$
    Update θ using $g \propto \nabla_\theta \sum_t \log \pi(a = r_t|s_t; \theta) R_{total}$
  end if
end for

node e_{source}, the agent picks a relation according to the stochastic policy $\pi(a|s)$, which is a probability distribution over all relations, to extend its reasoning path. This relation link may lead to a new entity, or it may lead to nothing. These failed steps will cause the agent to receive negative rewards. The agent will stay at the same state after these failed steps. Since the agent is following a stochastic policy, the agent will not get stuck by repeating a wrong step. To improve the training efficiency, we limit the episode length with an upper bound max_length. The episode ends if the agent fails to reach the target entity within max_length steps. After each episode, the policy network is updated using the following gradient:

$$\nabla_\theta J(\theta) = \nabla_\theta \sum_t \log \pi(a = r_t|s_t; \theta) R_{total} \tag{2.4}$$

where R_{total} is the linear combination of the defined reward functions. The detail of the retrain process is shown in Algorithm 1. In practice, θ is updated using the Adam Optimizer [44] with L_2 regularization.

2.3.3 Bi-directional Path-constrained Search

Given an entity pair, the reasoning paths learned by the RL agent can be used as logical formulas to predict the relation link. Each formula is verified using a bi-directional search. In a typical KG, one entity node can be linked to a large number of neighbors with the same relation link. A simple example is the relation $personNationality^{-1}$, which denotes the inverse of *personNationality*. Following this link, the entity *United States* can reach numerous neighboring entities. If the formula consists of such links, the number of intermediate entities can exponentially increase as we follow the reasoning formula. However, we observe that for these formulas, if we verify the formula from the inverse direction. The number of intermediate nodes can be tremendously decreased. Algorithm 2 shows a detailed description of the proposed bi-directional search.

2.4 Experiments

To evaluate the reasoning formulas found by our RL agent, we explore two standard KG reasoning tasks: link prediction (predicting target entities) and fact prediction (predicting whether an unknown fact holds or not). We compare our method with both path-based methods and embedding based methods. After that, we further analyze the reasoning paths found by our RL agent. These highly predictive paths validate the effectiveness of the reward functions. Finally, we conduct a experiment to investigate the effect of the supervised learning procedure.

Algorithm 2 Bi-directional search for path verification

Given a reasoning path $p : r_1 \rightarrow r_2 \rightarrow ... \rightarrow r_n$
for (e_i, e_j) in test set $\mathcal{D}$ **do**
    start $\leftarrow$ 0; end $\leftarrow$ n
    $left \leftarrow \emptyset; right \leftarrow \emptyset$
    while start $<$ end **do**
        $leftEx \leftarrow \emptyset; rightEx \leftarrow \emptyset$
        if len(left) $<$ len(right) **then**
            Extend path on the left side
            Add connected nodes to $leftEx$
            $left \leftarrow leftEx$
        else
            Extend path on the right side
            Add connected nodes to $rightEx$
            $right \leftarrow rightEx$
        end if
    end while
    if $left \cap right \neq \emptyset$ **then**
        return **True**
    elsereturn **False**
    end if
end for

2.4.1 Dataset and Settings

Dataset	# Ent.	# R.	# Triples	# Tasks
FB15K-237	14,505	237	310,116	20
NELL-995	75,492	200	154.213	12

Table 2.1: Statistics of the Datasets. # Ent. denotes the number of unique entities and # R. denotes the number of relations

Table 2.1 shows the statistics of the two datasets we conduct our experiments on. Both of them are subsets of larger datasets. The triples in FB15K-237 [32] are sampled from FB15K [25] with redundant relations removed. We perform the reasoning tasks on 20 relations which have enough reasoning paths. These tasks consists of relations from different domains like *Sports*, *People*, *Locations*, *Film*, etc. Besides, we present a new NELL subset that is suitable for multi-hop reasoning from the 995th iteration of the

NELL system. We first remove the triples with relation *generalizations* or *haswikipediaurl*. These two relations appear more than 2M times in the NELL dataset, but they have no reasoning values. After this step, we only select the triples with Top-200 relations. To facilitate path finding, we also add the inverse triples. For each triple (h, r, t), we append (t, r^{-1}, h) to the datasets. With these inverse triples, the agent is able to step backward in the KG.

For each reasoning task r_i, we remove all the triples with r_i or r_i^{-1} from the KG. These removed triples are split into train and test samples. For the link prediction task, each h in the test triples $\{(h, r, t)\}$ is considered as one query. A set of candidate target entities are ranked using different methods. For fact prediction, the true test triples are ranked with some generated false triples.

2.4.2 Baselines and Implementation Details

Most KG reasoning methods are based on either path formulas or KG embeddings. we explore methods from both of these two classes in our experiments. For path based methods, we compare our RL model with the PRA [24] algorithm, which has been used in a couple of reasoning methods [29, 35]. PRA is a data-driven algorithm using random walks (RW) to find paths and obtain path features. For embedding based methods, we evaluate several state-of-the-art embeddings designed for knowledge base completion, such as TransE [25], TransH [41], TransR [45] and TransD [46] .

The implementation of PRA is based on the code released by [24]. We use the TopK negative mode to generate negative samples for both train and test samples. For each positive samples, there are approximately 10 corresponding negative samples. Each negative sample is generated by replacing the true target entity t with a faked one $t^{'}$ in each triple (h, r, t). For TransE,R,H,D, we learn a separate embedding matrix for each

reasoning task using the positive training entity pairs. All these embeddings are trained for 1,000 epochs.[1]

Our RL model make use of TransE to get the continuous representation of the entities and relations. We use the same dimension as TransE, R to embed the entities. Specifically, the state vector we use has a dimension of 200, which is also the input size of the policy network. To reason using the path formulas, we adopt a similar linear regression approach as in PRA to re-rank the paths. However, instead of using the random walk probabilities as path features, which can be computationally expensive, we simply use binary path features obtained by the bi-directional search. We observe that with only a few mined path formulas, our method can achieve better results than PRA's data-driven approach.

2.4.3 Results

	FB15K-237			
Tasks	PRA	RL	TransE	TransR
teamSports	**0.987**	0.955	0.896	0.784
birthPlace	0.441	**0.531**	0.403	0.417
personNationality	**0.846**	0.823	0.641	0.720
filmDirector	0.349	**0.441**	0.386	0.399
filmWrittenBy	0.601	0.457	0.563	**0.605**
filmLanguage	0.663	**0.670**	0.642	0.641
tvLanguage	0.960	**0.969**	0.804	0.906
capitalOf	**0.829**	0.783	0.554	0.493
organizationFounded	0.281	0.309	**0.390**	0.339
musicianOrigin	0.426	**0.514**	0.361	0.379
Overall	0.541	**0.572**	0.532	0.540

Table 2.2: Link prediction results (MAP) on FB15K-237.

[1]The implementation we used can be found at `https://github.com/thunlp/Fast-TransX`

Tasks	NELL-995			
	PRA	RL	TransE	TransR
athletePlaysForTeam	0.547	**0.750**	0.627	0.673
athletePlaysInLeague	0.841	**0.960**	0.773	0.912
athleteHomeStadium	0.859	**0.890**	0.718	0.722
athletePlaysSport	0.474	0.957	0.876	**0.963**
teamPlaySports	0.791	0.738	0.761	**0.814**
orgHeadquaterCity	**0.811**	0.790	0.620	0.657
worksFor	0.681	**0.711**	0.677	0.692
bornLocation	0.668	0.757	0.712	**0.812**
personLeadsOrg	0.700	**0.795**	0.751	0.772
orgHiredPerson	0.599	**0.742**	0.719	0.737
Overall	0.675	**0.796**	0.737	0.789

Table 2.3: Link prediction results (MAP) on NELL-995.

Methods	Fact Prediction Results	
	FB15K-237	NELL-995
RL	**0.311**	**0.493**
TransE	0.277	0.383
TransH	0.309	0.389
TransR	0.302	0.406
TransD	0.303	0.413

Table 2.4: Fact prediction results (MAP) on two datasets.

Link Prediction This task is to rank the target entities given a query entity. Table 2.2 shows the mean average precision (MAP) results on two datasets. Since path-based methods generally work better than embedding methods for this task, we do not include the other two embedding baselines in this table. Instead, we spare the room to show the detailed results on each relation reasoning task.

For the overall MAP shown in the last row of the table, our approach significantly outperforms both the path-based method and embedding methods on two datasets, which validates the strong reasoning ability of our RL model. For most relations, since the em-

	# of Reasoning Paths	
Tasks	PRA	RL
worksFor	247	25
teamPlaySports	113	27
teamPlaysInLeague	69	21
athletehomestadium	37	11
organizationHiredPerson	244	9
...		
Average #	137.2	20.3

Table 2.5: Number of reasoning paths used by PRA and our RL model. *RL achieved better MAP with a more compact set of learned paths.*

bedding methods fail to use the path information in the KG, they generally perform worse than our RL model or PRA. However, when there are not enough paths between entities, our model and PRA can give poor results. For example, for the relation *filmWrittenBy*, our RL model only finds 4 unique reasoning paths, which means there is actually not enough reasoning evidence existing in the KG. Another observation is that we always get better performance on the NELL dataset. By analyzing the paths found from the KGs, we believe the potential reason is that the NELL dataset has more short paths than FB15K-237 and some of them are simply synonyms of the reasoning relations.

Fact Prediction Instead of ranking the target entities, this task directly ranks all the positive and negative samples for a particular relation. The PRA is not included as a baseline here, since the PRA code only gives a target entity ranking for each query node instead of a ranking of all triples. Table 2.4 shows the overall results of all the methods. Our RL model gets even better results on this task. We also observe that the RL model beats all the embedding baselines on most reasoning tasks.

Chapter 3

Few-Shot Learning for KB Completion

3.1 Introduction

To automatically complete KGs, the previous chapter (Chapter 2) and other studies [47, 25, 48, 49, 50, 35, 51, 52] have been made to build relational learning models that could infer missing triples by learning from existing ones. These methods explore the statistical information of triples or path patterns to infer new facts of existing relations; and have achieved considerable performance on various public datasets.

However, those datasets (*e.g.* FB15k, WN18) used by existing models mostly only cover common relations in KGs. For more practical scenarios, we believe the desired KG completion models should handle two key properties of KGs. A large portion of KG relations are actually long-tail. In other words, they have very few instances. The fewer training triples that one relation has, the more KG completion techniques could be of use. Therefore, it is crucial for models to be able to complete relations with limited numbers of triples. However, existing research usually assumes the availability of sufficient training

triples for all relations, which limits their usefulness on sparse long-tail relations.

Second, to capture up-to-date knowledge, real-world KGs are often dynamic and evolving at any given moment. New relations will be added whenever new knowledge is acquired. If a model can predict new triples given only a small number of examples, a large amount of human effort could be spared. However, to predict target relations, previous methods usually rely on well-learned representations of these relations. In the dynamic scenario, the representations of new relations cannot be sufficiently trained given limited training instances, thus the ability to adapt to new relations is also limited for current models.

In contrast to previous methods, here we present a model that depends only on the entity embeddings and local graph structures. Our model aims at learning a matching metric that can be used to discover more similar triples given one reference triple. The learnable metric model is based on a permutation-invariant network that effectively encodes the one-hop neighbors of entities, and also a recurrent neural network that allows multi-step matching. Once trained, the model will be able to make predictions about any relation while existing methods usually require fine-tuning to adapt to new relations. With two newly constructed datasets, we show that our model can achieve consistent improvement over various embedding models on the one-shot link prediction task.

3.2 Related Work

Embedding Models for Relational Learning Various models have been developed to model relational KGs in continous vector space and to automatically infer missing links. RESCAL [47] is one of the earlier work that models the relationship using tensor operations. [25] proposed to model relationships in the 1-D vector space. Following this line of research, more advanced models such as DistMult [48], ComplEx [49] and

ConvE [53] have been proposed. These embedding-based models usually assume enough training instances for all relations and entities and do not pay attention to those sparse symbols. More recently, several models [54, 55] have been proposed to handle unseen entities by leveraging text descriptions. In contrast to these approaches, our model deals with long-tail or newly added relations and focuses on one-shot relational learning without any external information, such as text descriptions of entities or relations.

Few-Shot Learning Recent deep learning based few-shot learning approaches fall into two main categories: (1) *metric based approaches* [56, 57, 58, 59], which try to learn generalizable metrics and the corresponding matching functions from a set of training tasks. Most methods in this class adopt the general matching framework proposed in deep siamese network [56]. One example is the Matching Networks [57], which make predictions by comparing the input example with a small labeled support set; (2) *meta-learner based approaches* [60, 61, 62, 63], which aim to learn the optimization of model parameters (by either outputting the parameter updates or directly predicting the model parameters) given the gradients on few-shot examples. One example is the LSTM-based meta-learner [60], which learns the step size for each dimension of the stochastic gradients. Besides the above categories, there are also some other styles of few-shot learning algorithms, *e.g.* Bayesian Program Induction [64], which represents concepts as simple programs that best explain observed examples under a Bayesian criterion.

Previous few-shot learning research mainly focuses on vision and imitation learning [65] domains. In the language domain, [59] proposed a multi-metric based approach for text classification. To the best of our knowledge, this work is the first research on few-shot learning for knowledge graphs.

3.3 Few-Shot Relational Learning

3.3.1 Problem Formulation

Knowledge graphs $\mathcal{G}$ are represented as a collection of triples $\{(h, r, t)\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, where $\mathcal{E}$ and $\mathcal{R}$ are the entity set and relation set. The task of knowledge graph completion is to either predict unseen relations r between two existing entities: $(h, ?, t)$ or predict the tail entity t given the head entity and the query relation: $(h, r, ?)$. As our purpose is to infer unseen facts for newly added or existing long-tail relations, we focus on the latter case. In contrast to previous work that usually assumes enough triples for the query relation are available for training, this work studies the case where only one training triple is available. To be more specific, the goal is to rank the true tail entity t_{true} higher than other candidate entities $t \in \mathcal{C}_{h,r}$, given only an example triple (h_0, r, t_0). The candidates set is constructed using the entity type constraint [32]. It is also worth noting that when we predict new facts of the relation r, we only consider a closed set of entities, *i.e.* no unseen entities during testing. For open-world settings where new entities might appear during testing, external information such as text descriptions about these entities are usually required and we leave this to future work.

3.3.2 One-Shot Learning Settings

This section describes the settings for the training and evaluation of our one-shot learning model.

The goal of our work is to learn a metric that could be used to predict new facts with one-shot examples. Following the standard one-shot learning settings [57, 60], we assume access to a set of training tasks. In our problem, each training task corresponds to a KG relations $r \in \mathcal{R}$, and has its own training/testing triples: $T_r = \{D_r^{train}, D_r^{test}\}$. This task

set is often denoted as the *meta-training* set, $\mathbb{T}_{meta-train}$.

To imitate the one-shot prediction at evaluation time, there is only one triple (h_0, r, t_0) in each D_r^{train}. The $D_r^{test} = \{(h_i, r, t_i, \mathcal{C}_{h_i,r})\}$ consists of the testing triples of r with ground-truth tail entities t_i for each query (h_i, r), and the corresponding tail entity candidates $\mathcal{C}_{h_i,r} = \{t_{ij}\}$ where each t_{ij} is an entity in $\mathcal{G}$. The metric model can thus be tested on this set by ranking the candidate set $\mathcal{C}_{h_i,r}$ given the test query (h_i, r) and the labeled triple in D_r^{train}. We denote an arbitrary ranking-loss function as $\ell_\theta(h_i, r, t_i|\mathcal{C}_{h_i,r}, D_r^{train})$, where θ represents the parameters of our metric model. This loss function indicates how well the metric model works on tuple $(h_i, r, t_i, \mathcal{C}_{h_i,r})$ while observing only one-shot data from D_r^{train}. The objective of training the metric model, *i.e.* the meta-training objective, thus becomes:

$$\min_\theta \mathbb{E}_{T_r} \left[\sum_{(h_i,r,t_i,\mathcal{C}_{h_i,r}) \in D_r^{test}} \frac{\ell_\theta(h_i, r, t_i|\mathcal{C}_{h_i,r}, D_r^{train})}{|D_r^{test}|} \right], \tag{3.1}$$

where T_r is sampled from the meta-training set $\mathbb{T}_{meta-train}$, and $|D_r^{test}|$ denotes the number of tuples in D_r^{test}.

Once trained, we can use the model to make predictions on new relations $r' \in \mathcal{R}'$, which is called the *meta-testing* step in literature. These meta-testing relations are unseen from meta-training, *i.e.* $\mathcal{R}' \cap \mathcal{R} = \phi$. Each meta-testing relation r' also has its own one-shot training data $D_{r'}^{train}$ and testing data $D_{r'}^{test}$, defined in the same way as in meta-training. These meta-testing relations form a meta-test set $\mathbb{T}_{meta-test}$.

Moreover, we leave out a subset of relations in $\mathbb{T}_{meta-train}$ as the meta-validation set $\mathbb{T}_{meta-validation}$. Because of the assumption of one-shot learning, the meta-testing relations do not have validation sets like in the traditional machine learning setting. Otherwise, the metric model will actually see more than one-shot labeled data during meta-testing, thus the one-shot assumption is violated.

Finally, we assume that the method has access to a *background knowledge graph* $\mathcal{G}'$,

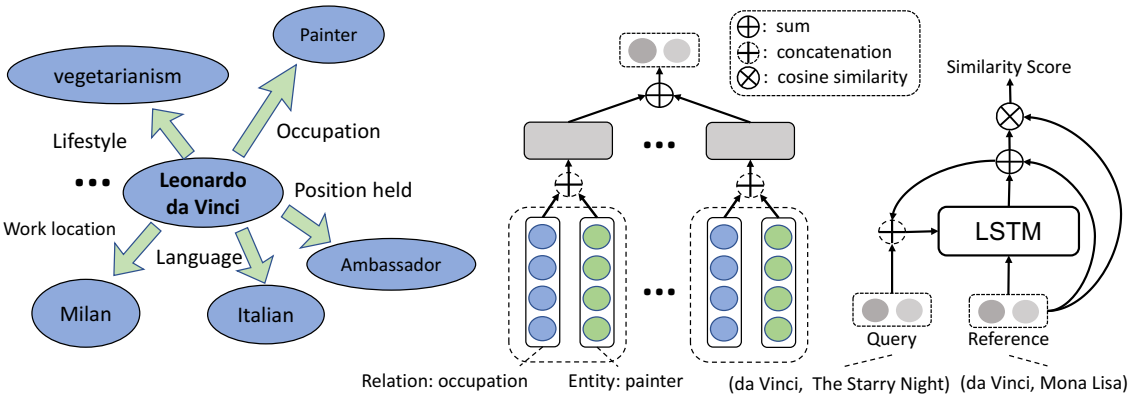


Figure 3.1: a) and b): Our **neighbor encoder** operating on entity *Leonardo da Vinci*; c): The **matching processor**.

which is a subset of $\mathcal{G}$ with all the relations from $\mathbb{T}_{meta-train}$, $\mathbb{T}_{meta-validation}$ and $\mathbb{T}_{meta-test}$ removed.

3.3.3 Model

In this section, we describe the proposed model for similarity metric learning and also the corresponding loss function ℓ we use to train our model.

The core of our proposed model is a similarity function $\mathcal{M}((h,t),(h',t')|\mathcal{G}')$. Thus for any query relation r, as long as there is one known fact (h_0,r,t_0), the model could predict the likelihood of testing triples $\{(h_i,r,t_{ij})|t_{ij} \in \mathcal{C}_{h_i,r}\}$, based on the matching score between each (h_i,t_{ij}) and (h_0,t_0). The implementation of the above matching function involves two sub-problems: (1) the representations of entity pairs; and (2) the comparison function between two entity-pair representations. Our overall model, as shown in Figure 3.1, deals with the above two problems with two major components respectively:

• **Neighbor encoder** (Figure 3.1b), aims at utilizing the local graph structure to better represent entities. In this way, the model can leverage more information that KG provides

for every entity within an entity pair.

• **Matching processor** (Figure 3.1c), takes the vector representations of any two entity pairs from the neighbor encoder; then performs multi-step matching between two entity-pairs and outputs a scalar as the similarity score.

3.3.4 Neighbor Encoder

This module is designed to enhance the representation of each entity with its local connections in knowledge graph.

Although the entity embeddings from KG embedding models [25, 48] already have relational information encoded, previous work [35, 66, 67] showed that explicitly modeling the structural patterns, such as paths, is usually beneficial for relationship prediction. In view of this, we propose to use a neighbor encoder to incorporate graph structures into our metric-learning model. In order to benefit from the structural information while maintaining the efficiency to easily scale to real-world large-scale KGs, our neighbor encoder only considers entities' local connections, *i.e.* the one-hop neighbors.

For any given entity e, its local connections form a set of (*relation, entity*) tuples. As shown in Figure 3.1a, for the entity *Leonardo da Vinci*, one of such tuples is (*occupation, painter*). We refer this neighbor set as as $\mathcal{N}_e = \{(r_k, e_k)|(e, r_k, e_k) \in \mathcal{G}'\}$. The purpose of our neighbor encoder is to encode $\mathcal{N}_e$ and output a vector as the latent representation of e. Because this is a problem of encoding sets with varying sizes, we hope the encoding function can be (1) invariant to permutations and also (2) insensitive to the size of the neighbor set. Inspired by the results from [68], we use the following function f that satisfies the above properties:

$$f(\mathcal{N}_e) = \sigma(\frac{1}{|\mathcal{N}_e|} \sum_{(r_k,e_k)\in\mathcal{N}_e} C_{r_k,e_k}). \tag{3.2}$$

where C_{r_k,e_k} is the feature representation of a relation-entity pair (r_k, e_k) and σ is the activation function. We set $\sigma = \tanh$ which achieves the best performance on $\mathbb{T}_{meta-validation}$.

To encode every tuple $(r_k, e_k) \in \mathcal{N}_e$ into C_{r_k,e_k}, we first use an embedding layer **emb** with dimension d (which can be pre-trained using existing embedding-based models) to get the vector representations of r_k and e_k:

$$v_{r_k} = \mathbf{emb}(r_k), v_{e_k} = \mathbf{emb}(e_k)$$

Dropout [69] is applied here to the vectors v_{r_k}, v_{e_k} to achieve better generalization. We then apply a feed-forward layer to encode the interaction within this tuple:

$$C_{r_k,e_k} = W_c(v_{r_k} \oplus v_{e_k}) + b_c, \tag{3.3}$$

where $W_c \in R^{d \times 2d}, b_c \in R^d$ are parameters to be learned and $\oplus$ denotes concatenation.

To enable batching during training, we manually specify the maximum number of neighbors and use all-zero vectors as "dummy" neighbors. Although different entities have different degrees (number of neighbors), the degree distribution is usually very concentrated, as shown in Figure 3.2. We can easily find a proper bound as the maximum number of neighbors to batch groups of entities.

The neighbor encoder module we propose here is similar to the Relational Graph Convolutional Networks [70] in the sense that we also use the shared kernel $\{W_c, b_c\}$ to encode the neighbors of different entities. But unlike their model that operates on the whole graph and performs multiple steps of information propagation, we only encode the local graphs of the entities and perform one-step propagation. This enables us to easily apply our model to large-scale KGs such as Wikidata. Besides, their model also does not operate on pre-trained graph embeddings. We leave the investigation of other graph

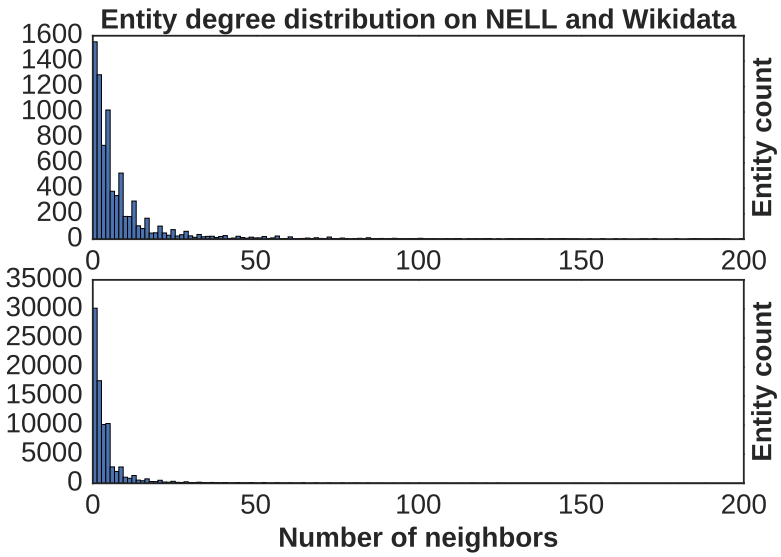


Figure 3.2: The distribution of entities' degrees (numbers of neighbors) on our two datasets. Since we work on closed-set of entities, we draw the figure by considering the intersection between entities in our background knowledge $\mathcal{G}'$ and the entities appearing in $\mathbb{T}_{meta-train}$, $\mathbb{T}_{meta-validation}$ or $\mathbb{T}_{meta-test}$. Note that all triples in $\mathbb{T}_{meta-train}$, $\mathbb{T}_{meta-validation}$ or $\mathbb{T}_{meta-test}$ are removed from $\mathcal{G}'$. Upper: NELL; Lower: Wikidata.

encoding strategies, e.g. [71, 72], to future work.

3.3.5 Matching Processor

Given the neighbor encoder module, now we discuss how we can do effective similarity matching based on our recurrent matching processor. By applying $f(\mathcal{N}_e)$ to the reference entity pair (h_0, t_0) and any query entity pair (h_i, t_{ij}), we get two neighbor vectors for each: $[f(\mathcal{N}_{h_0}); f(\mathcal{N}_{t_0})]$ and $[f(\mathcal{N}_{h_i}); f(\mathcal{N}_{t_{ij}})]$. To get a similarity score that can be used to rank (h_i, t_{ij}) among other candidates, we can simply concatenate the $f(\mathcal{N}_h)$ and $f(\mathcal{N}_t)$ in each pair to form a single pair representation vector, and calculate the cosine similarity between pairs. However, this simple metric model turns out to be too shallow and does not give good performance. To enlarge our model's capacity, we leverage a LSTM-based [73] recurrent "processing" block [74, 57] to perform multi-step matching. Every

Algorithm 3 One-shot Training

Input:
a) Meta-training task set $\mathbb{T}_{meta-training}$;
b) Pre-trained KG embeddings (excluding relation in $\mathbb{T}_{meta-training}$);
c) Initial parameters θ of the metric model;
for epoch = 0:M-1 **do**
    Shuffle the tasks in $\mathcal{T}_{meta-learning}$
    for T_r in $\mathcal{T}_{meta-learning}$ **do**
        Sample one triple as the reference
        Sample a batch B^+ of query triples
        Pollute the tail entity of query triples to get B^-
        Calculate the matching scores for triple in B^+ and B^-
        Calculate the batch loss $\mathcal{L} = \sum_B \ell$
        Update θ using gradient $g \propto \nabla\mathcal{L}$
    end for
end for

process step is defined as follows:

$$h^{'}_{k+1}, c_{k+1} = LSTM(q, [h_k \oplus s, c_k]) \tag{3.4}$$

$$h_{k+1} = h^{'}_{k+1} + q \tag{3.5}$$

$$score_{k+1} = \frac{h_{k+1} \odot s}{\|h_{k+1}\| \, \|s\|}, \tag{3.6}$$

where $LSTM(x, [h, c])$ is a standard LSTM cell with input x, hidden state h and cell state c, and $s = f(\mathcal{N}_{h_0}) \oplus f(\mathcal{N}_{t_0}), q = f(\mathcal{N}_{h_i}) \oplus f(\mathcal{N}_{t_{ij}})$ are the concatenated neighbor vectors of the reference pair and query pair. After K processing steps[1], we use $score_K$ as the final similarity score between the query and support entity pair. For every query $(h_i, r, ?)$, by comparing (h_i, t_{ij}) with (h_0, t_0), we can get the ranking scores for every $t_{ij} \in \mathcal{C}_{h_i,r}$.

[1] K is a hyperparameter to be tuned.

3.3.6 Loss Function and Training

For a query relation r and its reference/training triple (h_0, r, t_0), we collect a group of positive (true) query triples $\{(h_i, r, t_i^+)|(h_i, r, t_i^+) \in \mathcal{G}\}$ and construct another group negative (false) query triples $\{(h_i, r, t_i^-)|(h_i, r, t_i^-) \notin \mathcal{G}\}$ by polluting the tail entities. Following previous embedding-based models, we use a hinge loss function to optimize our model:

$$\ell_\theta = max(0, \gamma + score_\theta^- - score_\theta^+), \tag{3.7}$$

where $score_\theta^+$ and $score_\theta^-$ are scalars calculated by comparing the query triple $(h_i, r, t_i^+/t_i^-)$ with the reference triple (h_0, r, t_0) using our metric model, and the margin γ is a hyperparameter to be tuned. For every training episode, we first sample one task/relation T_r from the meta-training set $\mathbb{T}_{meta-training}$. Then from all the known triples in T_r, we sample one triple as the reference/training triple D_r^{train} and a batch of other triples as the positive query/test triples D_r^{test}. The detail of the training process is shown in Algorithm 3. Our experiments are discussed in the next section.

3.4 Experiments

3.4.1 Datasets

Dataset	# Ent.	# R.	# Triples	# Tasks
NELL-One	68,545	358	181,109	67
Wiki-One	4,838,244	822	5,859,240	183

Table 3.1: Statistics of the Datasets. # Ent. denotes the number of unique entities and # R. denotes the number of all relations. # Tasks denotes the number of relations we use as one-shot tasks.

Existing benchmarks for knowledge graph completion, such as FB15k-237 [32] and YAGO3-10 [75] are all small subsets of real-world KGs. These datasets consider the

same set of relations during training and testing and often include sufficient training triples for every relation. To construct datasets for one-shot learning, we go back to the original KGs and select those relations that do not have too many triples as one-shot task relations. We refer the rest of the relations as background relations, since their triples provide important background knowledge for us to match entity pairs.

Our first dataset is based on NELL [76], a system that continuously collects structured knowledge by reading webs. We take the latest dump and remove those inverse relations. We select the relations with less than 500 but more than 50 triples[2] as one-shot tasks. To show that our model is able to operate on large-scale KGs, we follow the similar process to build another larger dataset based on Wikidata [4]. The dataset statistics are shown in Table 3.1. Note that the Wiki-One dataset is an order of magnitude larger than any other benchmark datasets in terms of the numbers of entities and triples. For NELL-One, we use 51/5/11 task relations for training/validation/testing. For Wiki-One, the division ratio is 133:16:34.

3.4.2 Baselines and Implementation Details

In our experiments, we consider the following embedding-based methods: RESCAL [47], TransE [25], DistMult [48] and ComplEx [49]. For TransE, we use the code released by [45]. For the other models, we have tried the code released by [49] but it gives much worse results than TransE on our datasets. Thus we use our own implementations based on PyTorch [77] for comparison. When evaluating existing embedding models, during training, we use not only the triples of background relations but also all the triples of the training relations and the one-shot training triple of those validation/test relations. However, since the proposed metric model does not require the embeddings of query relations, we only include the triples of the background relations for embedding training.

[2]We want to have enough triples for evaluation.

As TransE and DistMult use 1-D vectors to represent entities and relations, they can be directly used in our natching model. While for RESCAL, since it uses matrices to represent relations, we employ mean-pooling over these matrices to get 1D embeddings. For the ComplEx model, we use the concatenation of the real part and imaginary part.

Apart from the above embedding models, a more recent method [53] applies convolution to model relationships and achieves the best performance on several benchmarks. For every query $(h, r, ?)$, their model enumerates the whole entity set to get positive and negative triples for training. We find that this training paradigm takes lots of computational resources when dealing with large entity sets and cannot scale to real-world KGs such as Wikidata that have millions of entities. For scalability issues, our experiments only consider models that use negative sampling for training.

3.4.3 Main Results

	NELL-One			
Model	MRR	Hits@10	Hits@5	Hits@1
RESCAL	.071/.140	.100/.229	.082/.186	.048/.089
TransE	.082/.093	.177/.192	.126/.141	.032/.043
DistMult	.075/.102	.128/.177	.093/.126	.045/.066
ComplEx	.072/.131	.128/.223	.041/.086	.041/.086
GMatching (RESCAL)	.144/**.188**	.277/.305	.216/.243	.087/**.133**
GMatching (TransE)	**.168**/.171	.293/.255	**.239**/.210	**.103**/.122
GMatching (DistMult)	.119/.171	.238/.301	.183/.221	.054/.114
GMatching (ComplEx)	.132/.185	**.308**/**.313**	.232/**.260**	.049/.119
GMatching (Random)	.083/.151	.211/.252	.135/.186	.024/.103
	Wiki-One			
Model	MRR	Hits@10	Hits@5	Hits@1
RESCAL	.119/.072	.167/.082	.132/.062	.093/.051
TransE	.023/.035	.036/.052	.029/.043	.015/.025
DistMult	.042/.048	.086/.101	.055/.070	.017/.019
ComplEx	079/.069	.148/.121	.106/.092	.046/.040
GMatching (RESCAL)	.113/.139	.330/.305	.180/.228	.033/.061
GMatching (TransE)	.167/.219	.349/.328	.289/.269	.083/.163
GMatching (DistMult)	.190/**.222**	**.384**/**.340**	**.291**/.271	.114/**.164**
GMatching (ComplEx)	**.201**/.200	.350/.336	.231/**.272**	**.141**/.120
GMatching (Random)	.174/.198	.309/.299	.222/.260	.121/.133

Table 3.2: Link prediction results on validation/test relations. KG embeddings baselines are shown at the top of the table and our one-shot learning (GMatching) results are shown at the bottom. **Bold** numbers denote the best results on meta-validation/meta-test. Underline numbers denote the model selection results from all KG embeddings baselines, or from all one-shot methods, i.e. selecting the method with the best validation score and reporting the corresponding test score.

The main results of our methods are shown in Table 3.2. We denote our method as "GMatching" since our model is trained to match local graph patterns. We use mean reciprocal rank (MRR) and Hits@K to evaluate different models. We can see that our method produces consistent improvements over various embedding models on these one-shot relations. The improvements are even more substantial on the larger Wiki-One dataset. To investigate the learning power of our model, we also try to train our metric model with randomly initialized embeddings. Surprisingly, although the results are worse than the metric models with pre-trained embeddings, they are still superior to the baseline embedding models. This suggests that, by incorporating the neighbor entities into our model, the embeddings of many relations and entities actually get updated in an effective way and provide useful information for our model to make predictions on test data.

It is worth noting that once trained, our model can be used to predict any newly added relations without fine-tuning, while existing models usually need to be re-trained to handle those newly added symbols. On a large real-world KG, this re-training process can be slow and highly computational expensive.

3.4.4 Ablation Studies

3.4.5 Analysis on Neighbor-Encoder

As our model leverages entities' local graph structures by encoding the neighbors, here we try to investigate the effect of the neighbor set by restricting the maximum number of neighbors. If the size of the true neighbor set is larger than the maximum limit, the neighbors are then selected by random sampling. Figure 3.3 shows the learning curves of different settings. These curves are based on the Hits@10 calculated on the validation set. We see that encoding more neighbors for every entity generally leads to better performance. We also observe that the model that encodes 40 neighbors in maximum

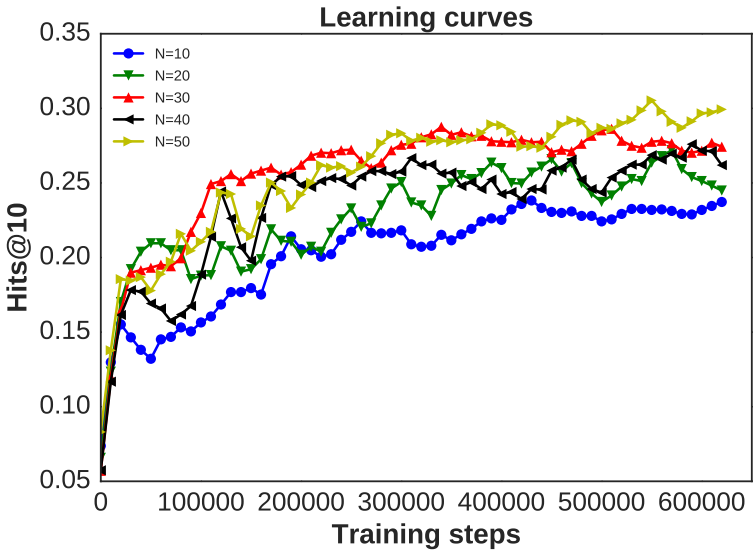


Figure 3.3: The learning curves on NELL-one. Every run uses different number of neighbors. The y-axis is Hits@10 calculated on all the validation relations.

actually yields worse performance than the model that only encodes 30 neighbors. We think the potential reason is that for some entity pairs, there are some local connections that are irrelevant and provide noisy information to the model.

We conduct ablation studies using the model that achieves the best Hits@10 on the NELL-One dataset. The results are shown in Table 3.3. We use Hits@10 on validation and test set for comparison, as the hyperparameters are selected using this evaluation metric. We can see that both the matching processor[3] and the neighbor encoder play important roles in our model. Another important observation is that the scaling factor $1/\mathcal{N}_e$ turns out to be very essential for the neighbor encoder. Without scaling, the neighbor encoder gives worse results compared to the simple embedding-based matching.

Configuration	Hits@10
Full Model with ComplEx	**.308/.313**
w/o Matching Processor	.266/.269
w/o Neighbor Encoder	.248/.296
w/o Scaling Factor	.229/.219

Table 3.3: Ablation on different components.

		MRR	
Relations	# Candidates	GMatching	ComplEx
sportsGameSport	123	0.424	**0.466**(0.139*)
athleteInjuredHisBodypart	299	0.025	0.026(**0.330***)
animalSuchAsInvertebrate	786	0.447	0.333(**0.555***)
automobilemakerDealersInCountry	1084	**0.453**	0.245(0.396*)
sportSchoolIncountry	2100	**0.534**	0.324(0.294*)
politicianEndorsesPolitician	2160	0.093	0.026(**0.194***)
agriculturalProductFromCountry	2222	**0.120**	0.029(0.042*)
producedBy	3174	0.085	0.040(**0.165***)
automobilemakerDealersInCity	5716	0.026	0.024(**0.041***)
teamCoach	10569	0.017	0.065(**0.376***)
geopoliticalLocationOfPerson	11618	0.028	0.016(**0.284***)

Table 3.4: Results decomposed over different relations. "*" denotes the results with standard training settings and "# Candidates" denotes the size of candidate entity set.

3.4.6 Discussion on Different Relations

When testing various models, we observe that the results on different relations are actually of high variance. Table 3.4 shows the decomposed results on NELL-One generated by our best metric model (GMatching-ComplEx) and its corresponding embedding method. For reference, we also report the embedding model's performance under standard training settings where 75% of the triples (instead of only one) are used for training and the rest are used for testing. We can see that relations with smaller candidate sets are generally easier and our model could even perform better than the embedding model

[3]Matching without Matching Processor is equivalent to matching using simple cosine similarity.

trained under standard settings. For some relations such as *athleteInjuredHisBodypart*, their involved entities have very few connections in KG. It is as expected that one-shot learning on these kinds of relations is quite challenging. Those relations with lots of (>3000) candidates are challenging for all models. Even for embedding model with more training triples, the performance on some relations is still very limited. This suggests that the knowledge graph completion task is still far from being solved.

3.5 Summary on KB Completion

Until this point, the techniques we have developed enable us to efficient improve the coverage of KBQA systems. The two aspects, i.e., interpretability and low-resource capability, are essential in real-world scenarios. However, there are still some important problems yet to be solved. For example, we have not explored a unified method that is take into account both aspects since this will be extremely challenging due to the sample complexity of RL algorithms. For future work, a structured meta learning algorithm which makes use of the semantic information of KB symbols might be a promising direction. Perhaps more importantly, the methods we developed here still assume a limited set of symbols and they are not able to adding knowledge about new entities or new relations. This limitation results from the inherent nature of knowledge bases and there is no straightforward way around unless we have more open-designed KBs. In the following chapters, we will start to look at using a more open set of knowledge source, i.e., a large-scale encyclopedia corpus. While directly using unstructured text data was believed to be impractical, powerful pretrained language models have began to show amazing promise and turn over this belief.

Chapter 4

Augment KBQA with Knowledge-Aware Text Reader

4.1 Introduction

In Chapter 2 and 3, we mainly focus on enriching the structured KBs themselves and leave the QA model untouched. Now, we explore an alternative model-based approach to improve the coverage of KBQA systems. We present a simple hybrid architecture that can ingest both structured and unstructured knowledge source to answer questions. Instead of relying on straightforward ensemble, the proposed model is able to make use of the KB information to improve the text reading.

With efficient text retrieval and complex cross-attention mechanims, text-based QA models along [17, 78, 79] have achieved remarkable performance when dealing with a single passage that is guaranteed to include the answer. However, they are still insufficient when multiple documents are presented. We hypothesize this is due to the lack of background knowledge while distinguishing relevant information from irrelevant ones (see Figure 4.1 for a real example).

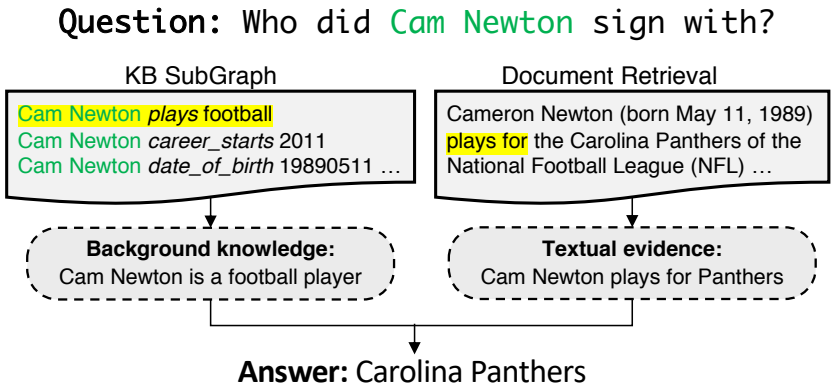


Figure 4.1: A real example from WebQSP. Here the answer cannot be directly found in the KB. But the knowledge provided by the KB, *i.e.*, *Cam Newton is a football player*, indicates he **signed with** *the team he plays for*. This knowledge can be essential for recognizing the relevant text piece.

To better utilize textual evidence for improving QA over incomplete KBs, this paper presents a new end-to-end model, which consists of (1) a simple yet effective subgraph reader that accumulates knowledge of each KB entity from a question-related KB subgraph; and (2) a knowledge-aware text reader that selectively incorporates the learned KB knowledge about entities with a novel conditional gating mechanism. With the specifically designed gate functions, our model has the ability to dynamically determine how much KB knowledge to incorporate while encoding questions and passages, thus is able to make the structured knowledge more compatible with the text information. Compared to the previous state-of-the-art [13], our model achieves consistent improvements with a much more efficient pipeline, which only requires a single pass of the evidence resources.

4.2 Task Definition

The QA task we consider here requires answering questions by reading knowledge base tuples $\mathcal{K} = \{(e_s, r, e_o)\}$ and retrieved Wikipedia documents $\mathcal{D}$. To build a scalable system, we follow [13] and only consider a subgraph for each question. The subgraph

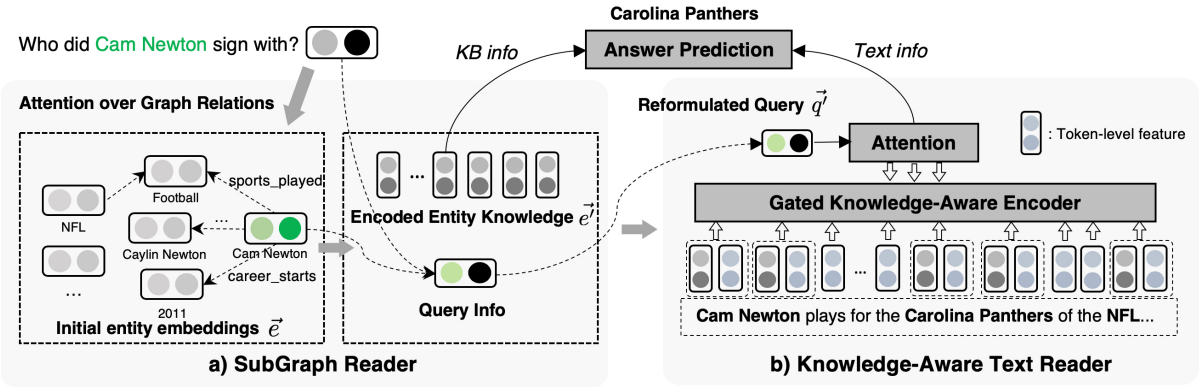


Figure 4.2: Model Overview. The subgraph reader **a**) first utilizes graph attention networks [80] to collect information for each entity in the question-related subgraph. The learned knowledge of each entity ($\vec{e'}$) is then passed to the text reader **b)** to reformulate the question representation ($\vec{q'}$) and encode the passage in a knowledge-aware manner. Finally, the information from the text and the KB subgraph is aggregated for answer entity prediction.

is retrieved by running Personalized PageRank [81] from the topic entities[1] (entities mentioned by the question: $\mathcal{E}_0 = \{e|e \in Q\}$). The documents $\mathcal{D}$ are retrieved by an existing document retriever [82] and further ranked by Lucene index. The entities in documents are also annotated and linked to KB entities. For each question, the model tries to retrieve answer entities from a candidate set including all KB and document entities.

4.3 QA with Hybrid Knowledge Source

The core components of our model consist of a graph-attention based KB reader (§4.3.1) and a knowledge-aware text reader (§4.3.2). The interaction between the modules is shown in Figure 4.2.

[1]Annotated by STAGG [10].

4.3.1 SubGraph Reader for KB Knowledge Extraction

This section describes the KB subgraph reader (SGReader), which employs graph-attention techniques to accumulate knowledge of each subgraph entity (e) from its linked neighbors (N_e). The graph attention mechanism is particularly designed to take into account two important aspects: (1) whether the neighbor relation is relevant to the question; (2) whether the neighbor entity is a topic entity mentioned by the question. After the propagation, the SGReader finally outputs a vectorized representation for each entity, encoding the knowledge indicated by its linked neighbors.

Question-Relation Matching To match the question and KB relation in an isomorphic latent space, we apply a shared LSTM to encode the question $\{w_1^q, w_2^q, ..., w_{l_q}^q\}$ and the tokenized relation $\{w_1^r, w_2^r, ..., w_{l_r}^r\}$. With the derived hidden states $\mathbf{h^q} \in \mathbb{R}^{l_q \times d_h}$ and $\mathbf{h^r} \in \mathbb{R}^{l_r \times d_h}$ for each word, we first compute the representation of relations with a self-attentive encoder:

$$\vec{r} = \sum_i \alpha_i \vec{h}_i^r, \ \alpha_i \propto \exp(\vec{w_r} \cdot \vec{h}_i^r), \tag{4.1}$$

where $\vec{h}_i^r$ is the i-th row of $\mathbf{h^r}$ and $\vec{w_r}$ is a trainable vector. Since a question needs to be matched with different relations and each relation is only described by part of the question, instead of matching the relations with a single question vector, we calculate the matching score in a more fine-grained way. Specifically, we first use $\vec{r}$ to attend each question token and then model the matching s_r by a dot product as follows:

$$s_r = \vec{r} \cdot \sum_j \beta_j \vec{h}_j^q, \beta_j \propto \exp(\vec{r} \cdot \vec{h}_j^q). \tag{4.2}$$

Extra Attention over Topic Entity Neighbors In addition to the question-relation similarities, we find another binary indicator feature derived from the topic entity is very useful. This indicator is defined as $I[e_i \in \mathcal{E}_0]$ for a neighbor (r_i, e_i) of an arbitrary entity e. Intuitively, if one neighbor links to a topic entity that appear in the question then the

corresponding tuple (e, r_i, e_i) could be more relevant than other non-topic neighbors for question answering. Formally, the final attention score $\tilde{s}_{(r_i,e_i)}$ over each neighbor (r_i, e_i) is defined as:

$$\tilde{s}_{(r_i,e_i)} \propto \exp(I[e_i \in \mathcal{E}_0] + s_{r_i}).$$

Information Propagation from Neighbors To accumulate the knowledge from the linked tuples, we define the propagation rule for each entity e:

$$\vec{e'} = \gamma^e \vec{e} + (1 - \gamma^e) \sum_{(e_i,r_i)\in N_e} \tilde{s}_{(r_i,e_i)} \sigma(\mathbf{W_e}[\vec{r_i}; \vec{e_i}]),$$

where $\vec{e}$ and $\vec{e_i}$ are pre-computed knowledge graph embeddings, $\mathbf{W_e} \in \mathbb{R}^{h_d \times 2h_d}$ is a trainable transformation matrix and $\sigma(\cdot)$ is an activation function. In addition, γ_e is a trade-off parameter calculated by a linear gate function as $\gamma^e = g(\vec{e}, \sum_{(e_i,r_i)\in N_e} \tilde{s}_{(r_i,e_i)} \sigma(\mathbf{W_e}[\vec{r_i}; \vec{e_i}]))$[2] , which controls how much information in the original entity representation should be retained.[3]

4.3.2 Knowledge-Aware Text Reader

With the learned KB embeddings, our model enhances text reading with KAReader. Briefly, we use an existing reading comprehension model [82] and improve it by learning more knowledge-aware representations for both question and documents.

Query Reformulation in Latent Space First, we update the question representation in a way that the KB knowledge of the topic entity can be incorporated. This allows the reader to discriminate relevant information beyond text matching.

Formally, we first take the original question encoding $\mathbf{h^q}$ and apply a self-attentive encoder to get a stand-alone question representation: $\vec{q} = \sum_i b_i \vec{h_i^q}$. We collect the topic

[2] $g(x, y) = \text{sigmoid}(\mathbf{W}[x; y]) \in (0, 1)$.

[3] The above step can be viewed as a gated version of the graph encoding techniques in NLP, e.g., [72, 71]. These general graph-encoders and graph-attention techniques may help when the questions require more hops and we leave the investigation to future work.

entity knowledge of the question by $\vec{e^q} = \sum_{e \in \mathcal{E}_0} \vec{e'}/|\mathcal{E}_0|$. Then we apply a gating mechanism to fuse the original question representation and the KB knowledge:

$$\vec{q'} = \gamma^q \vec{q} + (1 - \gamma^q) \tanh(\mathbf{W^q}[\vec{q}, \vec{e^q}, \vec{q} - \vec{e^q}]),$$

where $\mathbf{W^q} \in \mathbb{R}^{h_d \times 3h_d}$, and $\gamma^q = \text{sigmoid}(\mathbf{W^{gq}}[\vec{q}, \vec{e^q}, \vec{q} - \vec{e^q}])$ is a linear gate.

Knowledge-aware Passage Enhancement To encode the retrieved passages, we use a standard bi-LSTM, which takes several token-level features[4]. With the entity linking annotations in passages, we fuse the entity knowledge with the token-level features in a similar fashion as the query reformulation process. However, instead of applying a standard gating mechanism [83, 84], we propose a new conditional gating function that explicitly conditions on the question $\vec{q'}$. This simple modification allows the reader to dynamically select the inputs according to their relevance to the question. Considering a passage token w_i^d with its token features $\vec{f}_{w_i}^d$ and its linked entity e_{w_i}[5], we define the conditional gating function as:

$$\vec{i}_{w_i}^d = \gamma^d \vec{e'}_{w_i} + (1 - \gamma^d) \vec{f}_{w_i}^d, \text{ where}$$
$$\gamma^d = \text{sigmoid}(\mathbf{W^{gd}}[\vec{q} \cdot \vec{e'}_{w_i}; \vec{q} \cdot \vec{f}_{w_i}^d]).$$

$\vec{e'}_{w_i}$ denotes the entity embedding learned by our SGReader.

Entity Info Aggregation from Text Reading Finally we feed the knowledge-augmented inputs $\vec{i}_{w_i}^d$ into the biLSTM and use the output token-level hidden state $\vec{h}_{w_i}^d$ to calculate the attention scores $\lambda_i = \vec{q'}^T \vec{h}_{w_i}^d$. Afterwards, we get each document's representation as $\vec{d} = \sum_i \lambda_i \vec{h}_{w_i}^d$. For a certain entity e and all the documents containing

[4]We use the same set of features as in [82] except for the tagging labels.
[5]Non-entity tokens are encoded with token-level features only.

e: $\mathcal{D}^e = \{d|e \in d\}$, we simply aggregate the information by averaging the representations of linked documents as $\vec{e}_d = \frac{1}{|\mathcal{D}^e|}\sum_{d\in\mathcal{D}^e}\vec{d}$.

4.3.3 Answer Prediction

With entities representations ($\vec{e'}$ and $\vec{e^d}$), we predict the probability of an entity being the answer by matching the query vectors and the entity representations: $s^e = \sigma_s(\vec{q'}^T\mathbf{W_s}[\vec{e'};\vec{e^d}])$.

4.4 Experiments

Model	10% KB		30% KB		50% KB		100% KB	
	Hit@1	F1	Hit@1	F1	Hit@1	F1	Hit@1	F1
KV-KB	12.5	4.3	25.8	13.8	33.3	21.3	46.7	38.6
GN-KB	15.5	6.5	34.9	**20.4**	47.7	**34.3**	**66.7**	**62.4**
SGReader (Ours)	**17.1**	**7.0**	**35.9**	20.2	**49.2**	33.5	66.5	58.0
KV-KB+Text	24.6	14.4	27.0	17.7	32.5	23.6	40.5	30.9
GN-LF	29.8	17.0	39.1	25.9	46.2	35.6	65.4	56.8
GN-EF	31.5	17.7	40.7	25.2	49.9	34.7	67.8	60.4
Ours	**33.6**	**18.9**	**42.6**	**27.1**	**52.7**	**36.1**	67.2	57.3
GN-LF+EF (ensemble)	33.3	19.3	42.5	26.7	52.3	37.4	68.7	62.3

Table 4.1: Comparisons with Key-Value Memory Networks and GRAFT-Nets under different KB settings.

4.4.1 Setup

Dataset Our experiments are based on the WebQSP dataset [85]. To simulate the real-world scenarios, we test our models following the settings of [13], where the KB is downsampled to different extents. For a fair comparison, the retrieved document set is the same as the previous work.

Baselines and Evaluation Key-Value (KV) Memory Network [86] is a simple baseline that treats KB triples and documents as memory cells. Specifically, we consider its two variants, **KV-KB** and **KV-KB+Text**. The former is a KB-only model while the latter uses both KB and text. We also compare to the latest method **GraftNet** (GN) [13], which treats documents as a special genre of nodes in KBs and utilizes graph convolution [87] to aggregate the information. Similar to the KV-based baselines, we denote **GN-KB** as the KB-only version. Further, both **GN-LF** (late fusion) and **GN-EF** (early fusion) consider both KB and text. The former one considers KB and texts as two separate graphs, and then ensembles the answer scores. GN-EF is the existing best single model, which considers KB and texts as a single heterogeneous graph and aggregate the evidence to predict a single answer score for each entity. F1 and His@1 are used for evaluation since multiple correct answers are possible.

4.4.2 Results and Analysis

We show the main results of different incomplete KB settings in Table 4.1. For reference, we also show the results under full KB settings (*i.e.*, 100%, all of the required evidence is covered by KB). The row of SGReader shows the results of our model using only KB evidence. Compared to the previous KBQA methods (KV-KB and GN-KB), SGReader achieves better results in incomplete KB settings and competitive performance with the full KB. Here we do not compare with existing methods that utilize semantic parsing annotations [85, 88]. It is worth noting that SGReader only needs one hop of graph propagation while the compared methods typically require multiple hops.

Augmenting the SGReader with our knowledge-aware reader (KAReader) results in consistent improvements in the settings with incomplete KBs. Compared to other baselines, although our model is built upon a stronger KB-QA base model, it achieves

Model	Hit@1	F1
Full Model	46.8	28.1
- w/o query reformulation	44.4	27.6
- w/o knowledge enhancement	45.2	27.0
- w/o conditional knowledge gate	44.4	27.0

Table 4.2: Ablation on dev under the 30% KB setting.

the largest absolute improvement. It is worth mentioning that our model is still a single model, but it achieves competitive results to the existing ensemble model (GN-LF+EF). The results demonstrate the advantage of our knowledge-aware text reader.

Ablation Study To study the effect of each KAReader component, we conduct ablation analysis under the 30% KB setting (Table 4.2). We see that both query reformulation and knowledge enhancement are essential to the performance. Additionally, we find the conditional gating mechanism proposed in §4.3.2 is important. When replacing it with a standard gate function (see the row *w/o conditional knowledge gate*), the performance is even lower than the reader without knowledge enhancement, suggesting our proposed new gate function is crucial for the success of knowledge-aware text reading. The potential reason is that without the question information, the gating mechanism might introduce some irrelevant and misleading knowledge.

Part II

Efficient and Deep QA from Text

Chapter 5

Learning Dense Corpus Representations for Efficient Retrieval

5.1 Introduction

Starting from this chapter, we will explore answering open-domain questions directly from large-scale text corpus, which promises a much higher recall than KBQA systems. However, unlike querying structured databases, it is much more challenging to locate the relevant piece of text evidence from a huge number of unstructured text documents. While large pretrained models have achieved superior performance over a wide range NLP tasks, they typically need lots of computation cost and cannot scale to millions of documents at inference time. Existing open-domain QA systems are typically made of two essential components [82]. A *retrieval module* first retrieves a compact set of paragraphs from the whole corpus (such as Wikipedia) that includes millions of documents. Then a *reading module* is deployed to extract an answer span from the retrieved paragraphs.

Over the past few years, much of the progress in open-domain QA has been focusing on improving the reading module of the system, which only needs to process a small number of retrieved paragraphs. Specifically, improvements include stronger reading comprehension models [89, 90, 91, 92] and paragraph reranking models [93, 94] that assign more accurate relevance scores to the retrieved paragraphs. However, the performance is still bounded by the retrieval modules, which simply rely on traditional IR methods such as TF-IDF or BM25 [95]. These methods retrieve text solely based on n-gram lexical overlap and can fail on cases when deep semantic matching is required and when there are no common lexicons between the question and the target paragraph.

While neural models have proven effective at learning deep semantic matching between text pairs [96, 97, 98, 99], they usually require computing question-dependent paragraph encodings (*i.e.*, the same paragraph will have different representations when considering different questions), which is formidable considering space constraints and retrieval efficiency in practice. More recent studies [100, 101, 102] show that such a dilemma can be resolved with large-scale matching-oriented pretraining. These approaches use separate encoders for questions and paragraphs and simply model the matching between the question and paragraph using inner products of the output vectors. Thus, these systems only need to encode all paragraphs in a question-agnostic fashion, and the resulted dense corpus index could be fixed and reused for all possible questions. While achieving significant improvements over the BM25 baseline across a set of information-seeking QA datasets, existing pretraining strategies are highly sample-inefficient and typically require a large batch size (up to thousands), such that diverse and effective negative question-paragraph pairs could be included in each batch. When using a small batch size in our experiments, the model ceases to improve after certain updates. Given that a 12G GPU can only store around 10 samples with the BERT-base architecture at training time, the wider usage of these methods to corpora with different domains (*e.g.*, non-encyclopedic

web documents or scientific publications) is hindered given modest GPU hardware.

In this chapter, we propose a simple and sample-efficient method for pretraining dense corpus representations. We achieve stronger open-domain QA performance compared to an existing method [100] that requires 6 times more computation at training time. Besides, our method uses a much smaller batch size and can be implemented with only a small number of GPUs, *i.e.*, we use at most 4 TITAN RTX GPUs for all our experiments. In a nutshell, the proposed method first uses a pretrained sequence-to-sequence model to generate high-quality pretraining data instead of relying on heuristics to create pseudo question-paragraph pairs; for the training algorithm, we use clustering techniques to get effective negative samples for each pair and progressively update the clusters. Our method's efficacy is further validated through ablation studies, where we replicate existing methods that use the same amount of resources. For the downstream QA experiments, we carefully investigate different finetuning objectives and show the different configurations of the retrieval and span prediction losses have nontrivial effects on the final performance. We hope this analysis could save the efforts on trying out various finetuning strategies of future research that focus on improving the retrieval component of open-domain QA systems.

5.2 Related Work

The task of answering questions without specifying specific domains has been intensively studied since the earlier TREC QA competitions [103]. Studies in the early stage [104, 105, 106, 107] mostly rely on highly sophisticated pipelines and heterogeneous resources. Built on the recent advances in machine reading comprehension, [82] show that open-domain QA can be simply formulated as a reading comprehension problem with the help of a standard IR component that provides candidate paragraphs for

answer extraction. This two-stage formulation is simple yet effective to achieve competitive performance while using Wikipedia as the only knowledge resource.

Following this formulation, a couple of recent studies have proposed to improve the system using stronger reading comprehension models [90, 89], more effective learning objectives [108, 92, 109] or paragraph reranking models [93, 94, 110]. However, the retrieval components in these systems are still based on traditional inverted index methods, which are efficient but might fail when the target paragraph does not have enough lexicon overlap with the question.

In contrast to the sparse term-based features used in TF-IDF or BM25, dense paragraph vectors learned by deep neural networks [111, 112] can capture much richer semantics beyond the n-gram term features. To build effective paragraph encoders tailed for the paragraph retrieval in open-domain QA, more recent studies [100, 101, 102] propose to pretrain Transformer encoders [113] with objectives that simulate the semantic matching between questions and paragraphs. For instance, [100] uses the inverse cloze pretraining task to train a bi-encoder model to match a sentence and the paragraph in which the sentence belongs to. These approaches demonstrate promising performance but require a lot of resources for pretraining. The focus of this paper is to reduce the computational requirements of building an effective corpus index.

5.3 Progressively Pretrained Dense Corpus Representation

We begin by introducing the network architectures used in our retrieval and reading comprehension models. Next, we present how to generate high-quality question-paragraph pairs for pretraining and how we progressively train the retrieval model with

effective negative instances. Finally, we show how to finetune the whole system for QA.

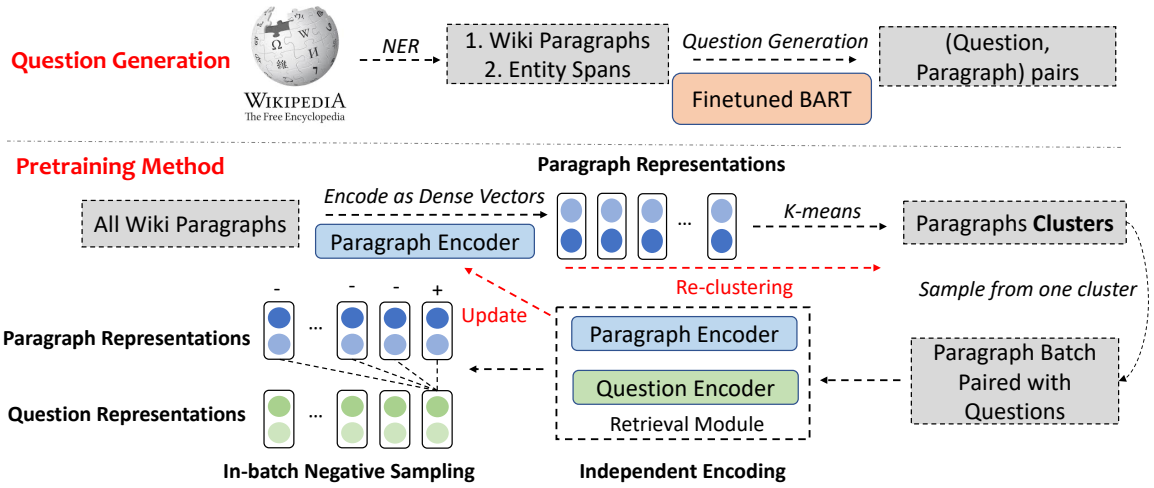


Figure 5.1: An overview of the progressive pretraining approach.

5.3.1 Model Architectures

Notations We introduce the following notations which will be used through our paper. The goal of open-domain QA is to find the answer derivation (p, s) from a large text corpus $\mathcal{C}$ given a question q, where p is an evidence paragraph and s is a text span within p. The start and end token of s are denoted as START(s) and END(s) respectively. We refer the retrieval module as $\mathrm{P}_\theta(p|q)$, with learnable parameters θ. Similarly, we refer the reading comprehension module as $\mathrm{P}_\phi(s|p, q)$, which can be decomposed as $\mathrm{P}_\phi(\mathrm{START}(s)|p, q) \times \mathrm{P}_\phi(\mathrm{END}(s)|p, q)$. We use D^k to represent the top-k paragraphs from the retrieval module; a subset of $D^* \in D^k$ represents the paragraphs in D^k that cover the correct answer; for each paragraph $p \in D^*$, we define S_p^* as all the spans in p that match the answer string.

The Retrieval Module We uses two isomorphic encoders to encode the questions and paragraphs, and the inner product of the output vectors is used as the matching

score. The encoders are based on the BERT-base architecture. We add linear layers $\mathbf{W}_q \in \mathbb{R}^{768\times128}$ and $\mathbf{W}_p \in \mathbb{R}^{768\times128}$ above the final representations of the [CLS] token to derive the question and paragraph representations:

$$h_q = \mathbf{W}_q \text{BERT}_Q(q)([\text{CLS}])$$
$$h_p = \mathbf{W}_p \text{BERT}_P(p)([\text{CLS}]),$$

The matching score is modeled as $h_q^\top h_p$. Thus, the probability of selecting p given q is calculated as:

$$\text{P}_\theta(p|q) = \frac{e^{h_q^\top h_p}}{\sum_{p'\in\mathcal{C}} e^{h_q^\top h_{p'}}}.$$

In practice, we only consider the top-k retrieved paragraphs $\mathcal{C}$ for normalization.

The Reading Module The architecture of our reading comprehension model is identical to the one in the original BERT paper [99]. We use two independent linear layers to predict the start and end position of the answer span. At training time, when calculating the span probabilities, we apply the shared-normalization technique proposed by [108], which normalizes the probability across all the top-k retrieved paragraphs. This encourages the model to produce globally comparable answer scores. We denote this probability as $\text{P}_\phi^{sn}(s|p,q)$ in contrast to the original formulation $\text{P}_\phi(s|p,q)$ that normalizes the probability within each paragraph.

5.3.2 The Pretrainining Method

We now describe how to pretrain the retrieval module using a better data generation strategy and a progressive training paradigm. Figure 5.1 depicts the whole pretraining process.

Pretraining Data Generation Previous dense retrieval approaches usually rely on simple heuristics to generate synthetic matching pairs for pretraining, which do not necessarily reflect the underlying matching pattern between questions and paragraphs. To minimize the gap between pretraining and the end task, we learn to generate high-quality questions from the paragraphs using a state-of-the-art pretrained seq2seq model, *i.e.*, BART [114]. More specifically, we finetune BART on the original NaturalQuestions dataset [115] such that it learns to generate questions given the groundtruth answer string and the groundtruth paragraph (labeled as *long answer* in NaturalQuestions). We concatenate the paragraph and the answer string with a separating token as the input to the BART model. We find this simple input scheme is effective enough to generate high-quality questions, achieving a 55.6 ROGUE-L score on the dev set. Afterward, we use spaCy[1] to recognize potential answer spans (named entities or dates) in all paragraphs in the corpus and use the finetuned BART model to generate the questions conditioned on the paragraph and each of the potential answers.

It is worth noting that the groundtruth answer paragraph supervision at this step could be eventually dropped and we could just use weakly supervised paragraphs to train the question generator; thus, our system becomes fully weak-supervised. As the pretraining process takes a long training time and lots of resources, we are unable to repeat the whole pretraining process with the weakly-supervised question generator. However, additional question generation experiments suggest that while using weakly-supervised paragraphs, the question generator still generates high-quality questions, achieving a ROUGE-L score of 49.6.[2]

[1] `https://spacy.io`

[2] For reference, a state-of-the-art QG model [116] trained with strong supervision achieves 49.9 ROUGE-L on a similar QA dataset also collected from real-user queries.

In-batch Negative Sampling To save computation and improve the sample efficiency of pretraining, we choose to use in-batch negative sampling [117] instead of gathering negative paragraphs for each question to pretrain the retrieval module. Specifically, for each pair (q, p) within a batch B, the paragraphs paired with other questions are considered negative paragraphs for q. Thus, the pretraining objective for each generated question is to minimize the negative log-likelihood of selecting the correct p among all paragraphs in the batch:

$$\mathcal{L}_{pre} = -\log \mathrm{P}_\theta(p|q). \tag{5.1}$$

A graphic illustration of this strategy is shown in Figure 5.1. As the batch size is usually very small compared to the number of all the paragraphs in the corpus, the pretraining task is much easier compared to the final retrieval task at inference time. In the whole corpus, there are usually lots of similar paragraphs and these paragraphs could act as strong distractors for each other in terms of both paragraph ranking and answer extraction. The desired retrieval model should be able to learn fine-grained matching instead of just learning to distinguish obviously different paragraphs. However, since existing dense retrieval methods typically use uniform batch sampling, there could be many easy negative samples in each batch, and they can only provide weak learning signals. Thus, a large batch size is usually adopted to include sufficient effective negative samples. Unfortunately, this is generally not applicable without hundreds of GPUs.

The Clustering-based Training Paradigm To provide effective negative samples, we propose a progressive training algorithm, as shown in the lower part of Figure 5.1. The key idea is to leverage the retrieval model itself to find groups of similar paragraphs. At a certain training step, we use the paragraph encoder at that moment to encode

Algorithm 4 The Clustering-based Progressive Pretraining

Input:
a) all (q, p) pairs from the question generation model;
b) the retrieval module BERT_Q and BERT_P;
while not finished **do**
    Encode the whole corpus with BERT_P;
    Clustering all paragraphs into C clusters using the dense encodings;
    for updates = 1:K **do**
        Random sample a paragraph cluster;
        Sample B paragraphs from the cluster;
        Fetch the corresponding questions;
        Calculate gradients wrt $\mathcal{L}_{pre}$;
        if updates % U == 0 **then**
            Update BERT_Q and BERT_P;
        end if
    end for
end while

the whole corpus and cluster all (q, p) pairs into many groups based on the similarity of their paragraph encodings. These groups are supposed to include similar paragraphs and potentially related questions. Then, we continue our pretraining by sampling each batch from one of the clusters. By doing this, we can provide challenging and effective negative paragraphs for each question, even with small batch size. Every time we recluster the whole corpus, the model will be encouraged to learn finer-grained matching between questions and paragraphs. Algorithm 4 provides a formal description of the entire process. Note that our training algorithm shares spirits with Curriculum Learning [118] and Self-Paced Learning [119, 120], in which the models are trained with harder instances as the training progresses. Instead of utilizing a predefined or dynamically generated order of all the instances according to their easiness, our algorithm makes use of a dynamic grouping of all the training instances and is specifically designed for the efficient in-batch negative sampling paradigm.

Method	Dataset		
	NaturalQuestions-Open	WebQuestions	CuratedTREC
DrQA [82]	-	20.7	25.7
R^3 [93]	-	17.1	28.4
DSQA [94]	-	25.6	29.1
HardEM [92]	28.1	-	-
PathRetriever [121]	32.6	-	-
WKLM [91]	-	34.6	-
GraphRetriever [122]	34.5	36.4	-
ORQA [100]	33.3	36.4	30.1
ProQA(Ours)	**37.4**	**37.1**	**34.6**

Table 5.1: Open-domain QA results in terms of exact answer match (EM). The first part of the table shows results from methods that use the traditional IR component. Note that these methods retrieve more paragraphs (typically dozens) than dense retrieval methods listed in the second part of the table, which only finds answers from the top-5.

5.3.3 QA Finetuning

Once pretrained, we use the paragraph encoder to encode the corpus into a large set of dense vectors. Following previous practice, we only finetune the question encoder and the reading module so that we can reuse the same dense index for different datasets. For every training question, we obtain the question representation h_q from the question encoder and retrieve the top-k paragraphs D^k on the fly using an existing maximum inner product search package. To train the reading module, we apply the shared-normalization trick and optimize the marginal probability of all matched answer spans in the top-k paragraphs:

$$\mathcal{L}_{reader} = -\log \sum_{p \in D^*} \sum_{s \in S_p^*} \mathrm{P}_\phi^{sn}(s|p,q). \tag{5.2}$$

In additional to the reader loss, we also incorporate the "early" loss used by [100], which updates the question encoder using the top-5000 dense paragraph vectors. If we define

D^*_{5000} as those paragraphs in the top-5000 that contain the correct answer, then the "early" loss is defined as:

$$\mathcal{L}_{early} = -\log \sum_{p \in D^*_{5000}} \mathrm{P}_\theta(p|q). \tag{5.3}$$

Thus our total finetuning loss is $\mathcal{L}_{early} + \mathcal{L}_{reader}$. Note this is different from the joint formulation used by [100] and [102], which consider the paragraphs as latent variables when calculating $\mathrm{P}(s|q)$. We find the joint objective does not bring additional improvements, especially after we use shared normalization. More variants of the finetuning objectives will be discussed in §5.4.5. At inference time, we use a linear combination of the retrieval score and the answer span score to rank the answer candidates from the top-5 retrieved paragraphs. The linear combination weight is selected based on the validation performance on each tested dataset.

5.4 Experiments

5.4.1 Datasets

We center our studies on QA datasets that reflect real-world information-seeking scenarios. We consider **1) NaturalQuestions-Open** [115, 100], which includes around 10K real-user queries (79,168/8,757/3,610 for train/dev/test) from Google Search; **2) WebQuestions** [123], which is originally designed for knowledge base QA and includes 5,810 questions (3,417/361/2,032 for train/dev/test) generated by Google Suggest API; **3) CuratedTREC** [124], which includes 2,180 real-user queries (1,353/133/694 for train/dev/test) from MSNSearch and AskJeeves logs. Compared to other datasets such as SQuAD [125] and TriviaQA [126], questions in these datasets are created without the

presence of ground-truth answers and the answer paragraphs, thus are less likely to have lexical overlap with the paragraph.

5.4.2 Implementation Details

For pretraining, we use a batch size of 80 and accumulate the gradients every 8 batches. We use the Adam optimizer [44] with learning rate 1e-5 and conduct 90K parameter updates. Following previous work [100], we use the 12-20-2018 snapshot of English Wikipedia as our open-domain QA corpus. When splitting the documents into chunks, we try to reuse the original paragraph boundaries and create a new chunk every time the length of the current one exceeds 256 tokens. Overall, we created 12,494,770 text chunks, which is on-par with the number (13M) reported in previous work. These chunks are also referred to as paragraphs in our work. For progressive training, we recluster all the chunks with k-means around every 20k updates using the paragraph encodings.

While finetuning the modules for QA, we fix the paragraph encoder in the retrieval module. For each question, we use the top-5 retrieved paragraphs for training and skip the question if the top-5 paragraphs fail to cover the answer. The MIPS-based retrieval is implemented with FAISS [127]. On NaturalQuestions-Open, we finetune for 4 epochs. To save the finetuning time on this large dataset, we only use a subset (2,000 out of 8,757) of the original development set for model selection. For WebQuestions and CuratedTREC (both of them are much smaller), we finetune for 10 epochs.

5.4.3 Answer Prediction Performance

Following existing studies, we use the exact match (EM) as the evaluation metric, which indicates the percentage of the evaluation samples for which the predicted span matches the groundtruth answers. In Table 5.1, we first show that our progressive method

Method	EM	model size	batch size	# updates
ORQA	33.3	330M	4096	100K
T5	36.6	11318M	-	-
REALM	**40.4**	330M	512	200K
ProQA	37.4	330M	80*8	90K

Table 5.2: Resource comparison with SOTA models. EM scores are measured on NaturalQuestions-Open. *batch size* and *updates* all refer to the dense index pretraining. Note that REALM uses ORQA to initialize its parameters and we only report the numbers after ORQA initialization. "80*8" indicates that we use a batch size of 80 and accumulate the gradients every 8 batches.

(denoted as **ProQA**) is superior to all of the open-domain QA systems (the upper part of the table) that use conventional IR methods, even though we only use the top-5 paragraphs to predict the answer while these methods use dozens of retrieved paragraphs. For the dense retrieval methods, we compare with ORQA [100], which is most relevant to our study but simply uses pseudo question-paragraph pairs for pretraining and also requires a larger batch size (4,096). We achieve much stronger performance than ORQA with much fewer updates and a limited number of GPUs. To the best of our knowledge, this is the first work showing that an effective dense corpus index can be obtained without using highly expensive computational resources. The reduced requirement of computation also makes our method easier to replicate for corpora in different domains.

In Table 5.2, we compare our method other more recently published QA systems in terms of both performance and computation cost. It is worth noting that although we need to use a BART model to generate training questions for Wikipedia documents, the inference cost of the question generator is still much lower than the training cost of our system and is not significant for comparing the overall computation cost: with the same GPU hardware, generating all the questions takes less than 1/6 of the training time. When using the *batch size*$\times$*# of updates* to approximate the training FLOPs, we see that our system is at least 6 times more efficient than ORQA. Compared to the recent proposed

T5 [128] approach, which converts the QA problem into a sequence-to-sequence (decode answers after encoding questions) problem and relies on the large model capacity to answer questions without retrieving documents, our system achieves better performance and is also much faster at inference time, due to the much smaller model size. The state-of-the-art REALM model [102] uses a more complicated pretraining approach that requires asynchronously refreshing the corpus index at train time. As it relies on ORQA initialization and further pretraining updates, it is even more computational expensive at training time. Also, as our method directly improves the ORQA pretraining, our method could easily stack with the REALM pretraining approach.

Concurrent to our work, [129] show that it is possible to use the groundtruth answer paragraphs in the original NaturalQuestions dataset to train a stronger dense retriever. However, they use a larger index dimension (768) while encoding paragraphs and also retrieve more paragraphs (20∼100) for answer extraction. As a larger index dimension naturally leads to better retrieval results [130] (despite sacrificing search efficiency) and using more paragraphs increases the recall of matched answer spans[3], this concurrent result is not directly comparable to ours, and we leave the combination effect of efficient pretraining and strong supervision (*i.e.*, using human-labeled paragraphs) to future work.

5.4.4 Ablation Studies

To validate the sample efficiency of our method, we replicate the inverse-cloze pretraining approach from ORQA using the same amount of resource as we used while training our model, *i.e.*, the same batch size and updates (90k). We also study the effect of the progressive training paradigm by pretraining the model with the same generated data but without the clustering-based sampling. We test the retrieval performance on

[3]While using more paragraphs, we achieve 40.6 EM (compared to 41.5 EM in [129]) even with a much smaller index dimension. Also, we did not use the gold paragraphs (strong supervision) to train our reading module.

Method	R@5	R@10	R@20
ProQA (90k)	**52.0**	**61.0**	**68.8**
ORQA⋆ (90k)	20.4	29.0	37.2
ProQA (no clustering, 90k)	42.9	52.6	60.8
ProQA (no clustering; 70k)	43.8	53.5	61.3
ProQA (no clustering; 50k)	38.8	48.2	56.7

Table 5.3: Ablation studies on different pretraining strategies. The retrieval modules (Recall@k) are tested on WebQuestions. ⋆Our reimplementation.

the WebQuestions test set before any finetuning. We use Recall@k as the evaluation metric, which measures how often the answer paragraphs appear in the top-k retrieval. The results are shown in Table 5.3. We can see that for the non-clustering version of our method, the improvements are diminishing as we reach certain training steps, while the progressive training algorithm brings around 8% improvements on different retrieval metrics. This suggests the importance of introducing more challenging negative examples in the batch when the batch size is limited. Comparing the no-clustering version of our method against ORQA, we see that using our data generation strategy results in much better retrieval performance (more than 22% improvements on all metrics).

5.4.5 Analysis on Finetuning Objectives

Noting that different finetuning configurations have been used in existing studies , we conduct additional experiments to investigate the efficacy of different finetuning objectives and provide insights for future research that intends to focus on improving the model itself. Specifically, we study the effects of using a joint objective [100, 102] and adding an additional reranking objective that is commonly used in sparse-retrieval QA systems [92, 91].

The joint objective treats the retrieved paragraphs as latent variables and optimizes

id	Objective Settings			EM
	joint	*rerank*	*shared-norm*	
1	-	-	✓	38.5
2	✓	-	✓	38.3
3	-	✓	✓	38.2
4	✓	-	-	36.2
5	-	-	-	35.1

Table 5.4: Analysis on different finetuning objectives on NaturalQuetions-Open. EM scores are measured on the 2,000 validation samples we used for model selection.

the marginal probability of the matched answer spans in all paragraphs:

$$\mathcal{L}_{joint} = -\log \sum_{p \in D^\star} \mathrm{P}_\theta(p|q) \sum_{s \in S_p^*} \mathrm{P}_\phi(s|p,q). \tag{5.4}$$

The reranking objective is usually implemented through a paragraph reranking module that uses a question-dependent paragraph encoder, *e.g.*, a BERT encoder that takes the concatenation of the question and paragraph as input. This kind of reranker has been shown to be beneficial to conventional IR methods since it can usually provide more accurate paragraph scores than the TF-IDF or BM25 based retriever while ranking the answer candidates. To implement this reranking module, we simply add another reranking scoring layer to our BERT-based span prediction module $\mathrm{P}_\phi(s|p,q)$, which encodes the paragraphs in a question-dependent fashion. At inference time, we use the paragraph scores predicted by this reranking component instead of the pretrained retrieval model to guide our final answer selection.

Table 5.4 shows the results of different objective settings. Comparing the results of (4) and (5), we can see that the joint objective can bring some improvements when shared-normalization is not applied. However, it does not yield improvements when shared-normalization is applied, according to the results of (1) and (2). By comparing (1) and (3), we see that with the strong pretrained retrieval model, adding an extra reranking

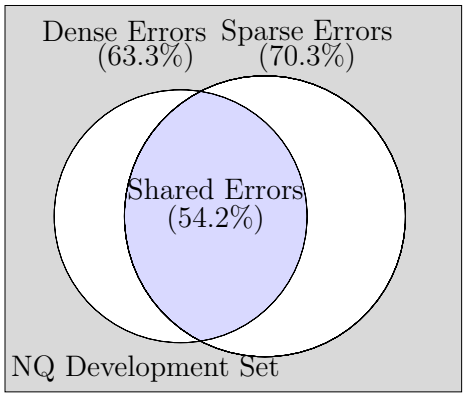


Figure 5.2: The error sets (from the official dev set of NaturalQuesitons-Open) of our dense retrieval method and a baseline system using sparse retrieval methods like TF-IDF. We use circles to represent the error sets of both systems. The percentages show the relative size of each set in terms of the whole dev set.

module that uses question-dependent paragraph encodings is no longer beneficial. This is partially because our pretrained retrieval model gets further improved during finetuning, in contrast to a fixed TF-IDF based retriever. Finally, from (1) and (5), we see that the shared normalization brings much larger improvements than the other factors. This aligns with the findings from an existing work [109] that only tested on SQuAD questions.

5.4.6 Error Analysis

To investigate the fundamental differences of the dense and sparse retrieval methods in open-domain QA, we conduct an error analysis using both the proposed method and a baseline system that uses TF-IDF and BM25 for retrieval. This baseline uses a similar retrieval pipeline as [92] and is trained with the finetuning objective defined in Eq. 5.2. This sparse retrieval baseline achieves 29.7 EM on the official dev set of NaturalQuestions-Open while our method achieves 36.7 EM[4]. Figure 5.2 shows the Venn diagram of the error sets from both systems. Our key findings are summarized in the following paragraphs.

[4]Note that this number is different from the number in Table 4 as we did not use the whole dev set for model selection.

The difference retrieval paradigms could complement each other. First, according to the error set differences (shown by the white regions in Figure 5.2), a considerable portion of the error cases (9.1% of the devt set) of our dense-retrieval system does not occur in the sparse-retrieval system and vice versa. This suggests the necessity of incorporating different retrieval paradigm when building real-world applications. In fact, the hybrid approach has already been adopted by a phrase-level retrieval method [131] and concurrent studies [130, 129].

Both systems are underestimated. As shown in Figure 5.2, 54.2% of the questions cannot be correctly answered by either system. However, our manual inspection on 50 of the shared error cases suggests that around 30% of these errors are due to annotation issues (14%) or the ambiguous nature of real-user queries (16%). One obvious annotation issue is the incompleteness of the answer labels. Example questions include *"When did Brazil lose to in 2014 World Cup?"*, to which both *"Germany"* and *"Netherlands"* are correct answers. This issue occurs because the annotators of NaturalQuestions only have a local view of the knowledge source as they are only asked to label the answer span using one document. In terms of the ambiguous questions, many of them are due to constraints unspecified by the question words, such as *"What is the population of New York City?"* (the time constraint is implicit) or *"When did Justice League come out in Canada?"* (needs entity disambiguation). This kind of questions result from the information-seeking nature of the open-domain QA task where the users usually use the minimal number of words for searching and they are not aware of the potential ambiguous factors. To solve this kind of questions, an interactive QA system might be necessary.

Chapter 6

Improving Machine Reading via Knowledge-Aware Pretraining

In this chapter, we focus on the answer extraction module of textual open-domain systems. Based on the observation that many information-seeking questions are highly entity-centric, we present a new complementary pretraining algorithm that explicitly emphasizes the aspect of entity knowledge.

6.1 Introduction

Over the past years, language models pretrained on a large amount of text such as ELMo [132]), BERT [133] and XLNet [134] have established new state of the art on a wide variety of NLP tasks. Researchers ascertain that pretraining allows models to learn syntactic and semantic information of language that is then transferred on other tasks [135, 136]. Interestingly, pretrained models also perform well on tasks that require grounding language and reasoning about the real world. For instance, the new state-of-the-art for WNLI [137], ReCoRD [138] and SWAG [139] is achieved by pretrained

models. These tasks are carefully designed so that the text input alone does not convey the complete information for accurate predictions – external knowledge is required to fill the gap. These results suggest that large-scale pretrained models implicitly capture real-world knowledge. [140] and [141] further validate this hypothesis through a zero-shot fact completion task that involves single-token entities, showing that pretrained models achieve much better performance than random guessing and can be on par with specifically-trained relation extraction models.

As unstructured text encodes a great deal of information about the world, large-scale pretraining over text data holds the promise of simultaneously learning syntax, semantics and connecting them with knowledge about the real world within a single model. However, existing pretraining objectives are usually defined at the token level and do not explicitly model entity-centric knowledge. Here, we investigate whether we can further enforce pretrained models to focus on encyclopedic knowledge about real-world entities, so that they can better capture entity information from natural language and be applied to improving entity-related NLP tasks. We evaluate the extent to which a pretrained model represents such knowledge by extending an existing fact completion evaluation to a cloze ranking setting that allows us to deal with a large number of multi-token entity names without manual judgments. Our experiments on 10 common Wikidata [4] relations reveal that existing pretrained models encode entity-level knowledge only to a limited degree. Thus, we propose a new weakly supervised knowledge learning objective that requires the model to distinguish between true and false knowledge expressed in natural language. Specifically, we replace entity mentions in the original documents with names of other entities of the same type and train the models to distinguish the correct entity mention from randomly chosen ones. Models trained with this objective demonstrates much stronger fact completion performance for most relations we test on. Compared with previous work [142, 143] that utilizes an external knowledge base to in-

corporate entity knowledge, our method is able to directly derive real-world knowledge from unstructured text. Moreover, our method requires no additional data processing, memory or modifications to the BERT model when fine-tuning for downstream tasks.

To test the model for downstream applications, we use four previously published datasets for open-domain QA and observe that questions in these datasets often concern entities. On three of the QA datasets, our pretrained model outperforms all previous methods that do not rely on memory-consuming inter-passage normalizations[1].

6.2 Related Work

Pretrained Language Representations Early research on language representations focused on static unsupervised word representations [145, 146]. Word embeddings leverage co-occurrences to learn latent word vectors that approximately reflect word semantics. Given that words can have different meanings in different contexts, more recent studies [147, 132] show that contextual language representations can be more powerful than static word embeddings in downstream tasks. This direction has been further explored at a larger scale with efficient Transformer architectures [148, 133, 134]. Our WKLM method is based on these techniques and we focus on improving the knowledge ability of pretrained models.

Knowledge-Enhanced NLP Models Background knowledge has been considered an indispensable part of language understanding [149, 150]. As standard language encoders usually do not explicitly model knowledge, recent studies [151, 152, 140, 153] have explored methods to incorporate external knowledge into NLP models. Most of these meth-

[1][144] propose to use BERT to encode multiple retrieved paragraphs within the same forward-backward pass. This operation requires around 12G GPU memory even for encoding a single example which includes 6 512-token paragraphs.

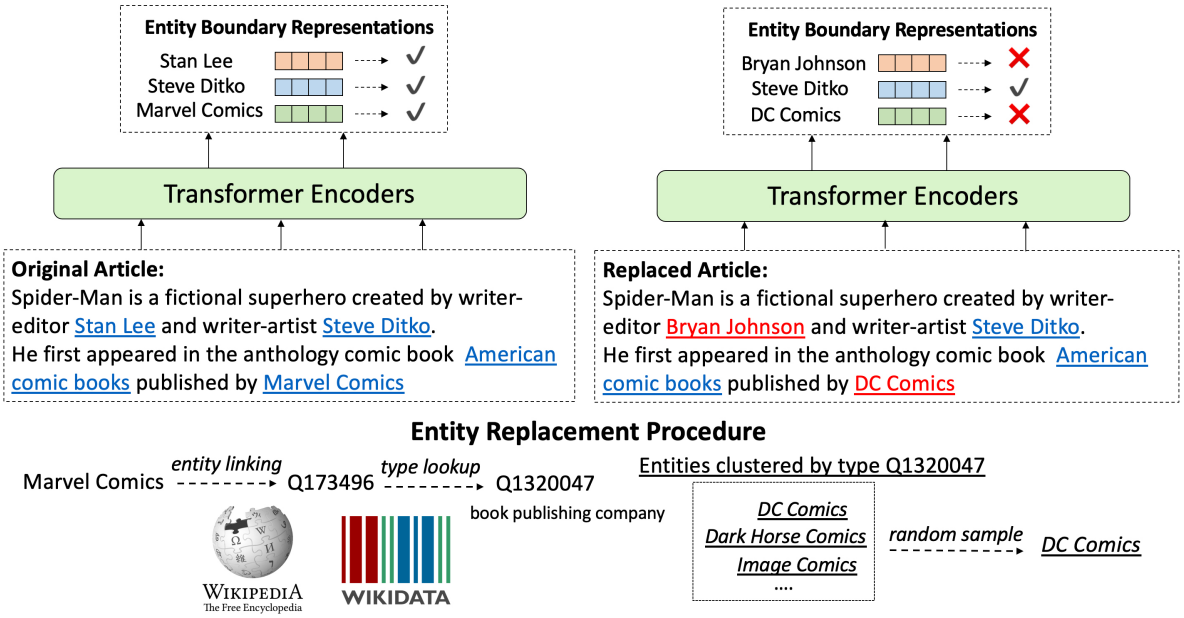


Figure 6.1: Type-Constrained Entity Replacements for Knowledge Learning.

ods rely on additional inputs such as entity representations from structured knowledge bases. With the breakthrough of large-scale pretrained language encoders [133], [142] and [143] adopt similar ideas and propose entity-level knowledge enhancement training objectives to incorporate knowledge into pretrained models. Other recent studies [154, 155] leverage external knowledge bases to enhance text-based question answering models. In contrast to these methods, our method utilizes minimal external entity information and does not require additional memory or architectural changes when applied to downstream tasks.

6.3 Entity Replacement Training

We design an entity-centric training objective that utilizes weakly supervised training signals to explicitly encourage knowledge learning during pretraining. Given an input

document, we first recognize the entity mentions and link them to Wikipedia entities[2]. We consider the original texts as positive knowledge statements and create negative statements by randomly replacing the entity mentions ($\mathcal{E}^+$) with the names of other random entities ($\mathcal{E}^-$) that have the same entity type as the mentioned entity. This setup is similar in spirit to the type-constrained negative sampling technique used to train knowledge base representations [25]. The latter technique creates negative triples by replacing the subject or object entity with random entities of the same type. Instead of knowledge base triples, we treat unstructured texts as factual statements. For a certain entity e mentioned in a context $\mathcal{C}$, we train the model to make a binary prediction indicating whether the entity has been replaced.

$$J_{e,\mathcal{C}} = 1_{e \in \mathcal{E}^+} \log P(e|\mathcal{C}) + (1 - 1_{e \in \mathcal{E}^+}) \log(1 - P(e|\mathcal{C})).$$

Compared to the language modeling objective, entity replacement is defined at the entity level and introduces stronger negative signals. When we enforce entities to be of the same type, we preserve the linguistic correctness of the original sentence while the system needs to learn to perform judgment based on the factual aspect of the sentence.

We describe the implementation in more detail in the following paragraphs.

Data Preparation We use the whole English Wikipedia dump as training data and rely on all Wikipedia entities[3]. Entities in documents are recognized based on Wikipedia anchor links and entity alias from Wikidata. That is, we first retrieve the entities annotated by anchor links and then find other mentions of these entities by string matching their Wikidata alias. We split each document into multiple text chunks with the same size (512 tokens). Although our experiments rely on the Wikipedia corpus, this setup

[2]The entity links are only required for pretraining.

[3]Each Wikipedia entity (title) corresponds to a unique entity node in Wikidata.

can be easily extended to larger corpora with off-the-shelf entity linking tools. We leave the larger scope of the experiments to future work.

Replacement Strategy When replacing entities, we first lookup type information[4] from Wikidata and then randomly select other entities with the same type. We do not replace adjacent entities. In other words, there must be at least one unreplaced entity between any two replaced ones. This reduces cases where we replace all entities in the same sentence and the resulting sentences happen to introduce correct entities by chance. For replacement, we randomly sample a string from the entities' alias set. For each text chunk, we replicate it 10 times with different negative entities for each replacement location. We show an illustration of the entity replacement method in Figure 6.1.

Model Architecture We use the Transformer [156] model used by BERT [133]. We use the same architecture as BERT base: 12 Transformer layers, each with hidden dimension 768. We initialize the transformer with a model pretrained based on our own BERT re-implementations[5]. For each entity, we use the final representations of its boundary words (words before and after the entity mention) to make predictions. We simply concatenate the boundary words' representations and add a linear layer for prediction. During training, we use 0.05 dropout at the final layer.

Training Objectives Masked language model pretraining has been proven to be effective for downstream tasks. While training for entity replacement we also train with the masked language model objective in a multi-task set-up. When masking tokens, we restrict the masks to be outside the entity spans. We use a masking ratio of 5% instead

[4]Based on the "instance of" relation in Wikidata. If there are multiple correct types, we randomly choose one.

[5]We use the masked language model implementation in Fairseq [157] to pre-train model for 2M updates on the combination of BooksCorpus [158] and English Wikipedia.

of 15% in the original BERT to avoid masking out too much of the context. We train the model for approximately 1 million updates using a batch size of 128.

6.4 Experiments

We first test our model on a fact completion task. This task resembles traditional knowledge base completion: it requires the model to complete missing entities in factual triples. We further test on two real-world downstream tasks that require entity-level knowledge – question answering and fine-grained entity typing.

6.4.1 Zero-Shot Fact Completion

In traditional knowledge base completion tasks models have access to a set of training triples. Instead, we utilize a zero-shot test to examine the model's ability to automatically derive relational knowledge from natural language.

Dataset We rely on factual triples from Wikidata. Each triple describes the relationship between two certain entities, *e.g.*, {*Paris, CapitalOf, France*}. Following recent practices [159, 140] that decode structured knowledge from language models, we first manually create templates to convert triples of 10 common relations into natural language expressions ({*Paris, CapitalOf, France*} $\rightarrow$ *the capital of France is Paris*). We then create queries by removing the object entity in the expression and use pre-trained models to predict the missing entities, *e.g.*, *the capital of France is ?*. We create 1000 cloze examples[6] for each of the 10 relations.

[6]For each relation, we use the top triples that connecting most common entities.

Evaluation Metrics Previous work [140, 141] either relies on human evaluation or only considers single-token entities for fact completion. In contrast, we consider an entity-ranking setup and create a set of candidate entities for each relation. This setting allows us to automatically evaluate a large number of queries that usually involve multi-token entities. We test pretrained models on their ability to recover the correct object entity from the candidate set. To create the negative choices, we select from the set of all object entities in the particular relation, which generally have the same type as the groundtruth and are more challenging to distinguish than entities with different types. Our evaluation strategy is similar to previous work on knowledge base completion [47, 25, 160]. We follow these studies and use Hits@10 as the evaluation metric.

Baselines We compare our model with two pretrained language models BERT [133] (both base and large) and GPT-2 [148]. We make use of their output token probabilities to rank candidate entities. For BERT, we feed in the masked queries (*e.g.*, Q_{masked} = `the capital of France is [MASK]`). For multi-token candidates, we use the same number of `[MASK]` tokens in the query inputs. We use the average log probability of masked tokens for ranking. Given a multi-token entity $E_i = [e_i^1, e_i^2, ..., e_i^{|E_i|}]$, the ranking score from BERT is calculated as

$$S_{E_i} = \frac{1}{|E_i|} \sum_k \log P(e_i^k | Q_{masked}).$$

For GPT-2, we feed in the original query without the answer entity and use the first-token probability of candidate entities for ranking, which performs better than using average log probabilities. As our model learns to predict a plausible probability ($P(e|\mathcal{C})$) for each entity mention during entity replacement training, we can directly use these predicted probabilities to rank the candidates.

Results Table 6.1 shows the fact completion results for all relations. We denote our method **WKLM** for (**W**eakly Supervised **K**nowledge-Pretrained **L**anguage **M**odel). Overall, WKLM achieves the best results on 8 of the 10 relations. We also observe that GPT-2 outperforms BERT on average. We think this is because the fact completion task requires models to predict the missing entities using only a short context on the left, while BERT pretraining incorporates context from both directions. Interestingly, BERT achieves good performance on several geographical relations such as `PlaceOfBirth`, `LocatedIn` and `PlaceOfDeath`. We conjecture that this is because location entities usually appear at sentence ends in Wikipedia articles, *e.g.*, `Obama was born in Honolulu, Hawaii.`. This sentence pattern is similar to our templates and BERT may learn to rely mostly on the left context to make predictions. For most relations that include answers that are person names, BERT lags behind both GPT-2 and our model.

Comparing the top and bottom five relations, we observe that BERT's performance is correlated with the size of the candidate set, while WKLM and GPT-2 are less sensitive to this number. A similar pattern exists between models' performance and the cardinality of groundtruth answers, *i.e.*, our model achieves similar performance on both single-answer and multiple-answer queries while BERT is usually better at single-answer queries. WKLM both outperforms BERT and GPT-2 and achieves robust performance across relations with different properties.

6.4.2 Downstream Tasks

Background knowledge is important for language understanding. We expect our pretraining approach to be beneficial to NLP applications where entity-level knowledge is essential. We consider two such applications: question answering and entity-typing.

Table 6.1: Zero-Shot Fact Completion Results.

Relation Name	# Options	Model			
		BERT-base	**BERT-large**	**GPT-2**	**Ours**
HasChild (P40)	906	9.00	6.00	20.5	**63.5**
NotableWork (P800)	901	1.88	2.56	2.39	**4.10**
CapitalOf (P36)	820	1.87	1.55	15.8	**49.1**
FoundedBy (P112)	798	2.44	1.93	8.65	**24.2**
Creator (P170)	536	4.57	4.57	7.27	**9.84**
PlaceOfBirth (P19)	497	19.2	**30.9**	8.95	23.2
LocatedIn (P131))	382	13.2	52.5	21.0	**61.1**
EducatedAt (P69)	374	9.10	7.93	11.0	**16.9**
PlaceOfDeath (P20)	313	**43.0**	42.6	8.83	26.5
Occupation (P106)	190	8.58	**10.7**	9.17	**10.7**
Average Hits@10	-	11.3	16.1	16.3	**28.9**

We find that a large portion of the questions in existing QA datasets are about entities and involve entity relations. In a way, our pretraining objective is analogous to question answering in a multiple-choice setting [14]. The entity-typing task requires the model to predict a set of correct types of entity mentions in a short context. The context itself can be insufficient and the training data for this task is small and noisy. We believe a model that encodes background entity knowledge can help in both cases.

Question Answering

Datasets We consider four question answering datasets:

- **WebQuestions** [161] is originally a dataset for knowledge base question answering. The questions are collected using Google Suggest API and are all asking about simple relational facts of Freebase entities.

- **TriviaQA**[7] [163] includes questions from trivia and quiz-league websites. Apart

[7]The splits of TriviaQA might be different in previous work, we use the same splits used by [162]. Different methods also use different retrieval systems for this dataset, *i.e.*, ORQA and BERTserini retrieve documents from Wikipedia while the other methods use retrieved Web documents.

from a small portion of questions to which the answers are numbers and free texts, 92.85% of the answers are Wikipedia entities.

- **Quasar-T** [164] is another dataset that includes trivia questions. Most of the answers in this dataset are none phrases. According to our manual analysis on random samples, 88% of the answers are real-world entities[8].

- **SearchQA** [165] uses questions from the television quiz show *Jeopardy!* and we also find that almost all of the answers are real-world entities.

Questions in all three datasets are created without the context of a paragraph, which resembles the scenario of practical question answering applications. All the questions except WebQuestions are written by humans. This indicates that humans are generally interested to ask questions to seek information about entities. We show the statistics and example questions in Table 6.2. We split the training data (created by distant supervision) of WebQuestions with a ratio (9:1) for training and development. Since our model is based on our own BERT implementations, in addition to the aforementioned entity-related datasets, we first use the standard SQuAD [16] benchmark to validate our model's answer extraction performance.

Table 6.2: Properties of the QA Datasets.

Dataset	**Train**	**Valid**	**Test**	**Example Questions**
WebQuestions	3778	-	2032	*Who plays Stewie Griffin on Family Guy?*
TriviaQA	87291	11274	10790	*What is the Japanese share index called?*
SearchQA	99811	13893	27247	*Hero several books 11 discover's wizard?*
Quasar-T	37012	3000	3000	*Which vegetable is a Welsh emblem?*

Settings We adopt the fine-tuning approach to extract answer spans with pretrained models. We add linear layers over the last hidden states of the pretrained models to

[8]We consider answers as entities as long as they correspond to Wikidata entities or Wikipedia titles.

predict the start and end positions of the answer. Unlike SQuAD, questions in the datasets we use are not paired with paragraphs that contain the answer. We follow previous work [82, 166] and retrieve context paragraphs with information retrieval systems. Details of the context retrieval process for each dataset can be found in the appendix. Reader models are trained with distantly supervised data, *i.e.*, we treat any text span in any retrieved paragraph as ground truth as long as it matches the original answers. Since the reader model needs to read multiple paragraphs to predict a single answer at inference time, we also train a BERT based paragraph ranker with distant-supervised data to assign each paragraph a relevance score. The paragraph ranker takes question and paragraph pairs and predicts a score in the range $[0, 1]$ for each pair. During inference, for each question and its evidence paragraph set, we first use the paragraph reader to extract the best answer from each paragraph. These answers are then ranked based on a linear combination of the answer extraction score (a log sum of the answer start and end scores) and the paragraph relevance score. We also evaluate model performance without using the relevance scores.

Open-Domain QA Baselines We compare our QA model with the following systems:

- **DrQA** [82] is an open-domain QA system which uses TF-IDF with bigram features for ranking and a simple attentive reader for answer extraction.
- **R**^{3} [166] is a reinforcement learning based system which jointly trains a paragraph ranker and a document reader.
- **DSQA** [162] uses RNN-based paragraph ranker and jointly trains the paragraph ranker and attentive paragraph ranker with a multi-task loss.
- **Evidence Aggregation** [167] uses a hybrid answer reranking module to aggregate answer information from multiple paragraphs and rerank the answers extracted

from multiple paragraphs.

- **BERTserini** [168] is a BERT-based open-domain QA system, which uses BM25-based retriever to retrieve 100 paragraphs and a BERT-based reader to extract answers. The paragraph reader is either trained with SQuAD [16] data or distant-supervision data [169]

- **ORQA** [170] replaces the traditional BM25 ranking with a BERT-based ranker. The ranker model is pretrained on the whole Wikipedia corpus with an inverse cloze task which simulates the matching between questions and paragraphs. All text blocks in Wikipedia are be pre-encoded as vectors and retrieved with Locality Sensitive Hashing.

Results Table 6.3 shows the SQuAD results and Table 6.4 shows the open-domain results on the four datasets that are highly entity-related. From the SQuAD results, we observe that our BERT reimplementation performs better than the original model this is due to the fact that it is trained for twice as many updates: 2 million vs. 1 million for the original BERT. Although lots of the answers in SQuAD are non-entity spans, the WKLM model we propose achieves better performance than BERT. We believe the improvement is due to both the masked language model and entity replacement objectives. Ablation experiments on the training objectives will be discussed in table 6.5.

Table 6.3: SQuAD Dev Results.

Model	EM	F1
Google's BERT-base	80.8	88.5
Google's BERT-large	84.1	90.9
Our BERT-base	83.4	90.5
WKLM (base)	84.3	91.3

Having established that our BERT re-implementation performs better than the origi-

nal model, we compare with only our own BERT for the following experiments. From Table 6.4, we see that our model produces consistent improvements across different datasets. Compared to the 0.8 F1 improvements over BERT on SQuAD, we achieve an average of 2.7 F1 improvements over BERT on entity-related datasets when the ranking scores are not used. On TriviaQA and Quasar-T, WKLM outperforms our BERT even when it uses ranking scores. Improvements in natural language question datasets (WebQuestions, TriviaQA, and Quasar-T) are more significant than SearchQA where the questions are informal queries. When we utilize ranking scores from a simple BERT based ranker, we are able to achieve the state-of-the-art on three of the four datasets.

Table 6.4: Open-domain QA Results.

Model	**WebQuestions**		**TriviaQA**		**Quasar-T**		**SearchQA**	
	EM	F1	EM	F1	EM	F1	EM	F1
DrQA [82]	20.7	-	-	-	-	-	-	-
R^3 [166]	-	-	50.6	57.3	42.3	49.6	57.0	63.2
DSQA [162]	18.5	25.6	48.7	56.3	42.2	49.3	49.0	55.3
Evidence Agg. [167]	-	-	50.6	57.3	42.3	49.6	57.0	63.2
BERTserini [168]	-	-	51.0	56.3	-	-	-	-
BERTserini+DS [169]	-	-	54.4	60.2	-	-	-	-
ORQA [170]	**36.4**	-	45.0	-	-	-	-	-
BERT	29.2	35.5	48.7	53.2	40.4	46.1	57.1	61.9
BERT+Ranking score	32.2	38.9	52.1	56.5	43.2	49.2	60.6	65.9
WKLM	30.8	37.9	52.2	56.7	43.7	49.9	58.7	63.3
WKLM+Ranking score	34.6	41.8	**58.1**	**63.1**	**45.8**	**52.2**	**61.7**	**66.7**

Ablation Study: The Effect of Masked Language Model Loss

In view of a recent study [171] showing simply extending the training time of BERT leads to stronger performance on various downstream tasks, we conduct further analysis to differentiate the effects of entity replacement training and masked language modeling. We compare our model with three variants: a model pretrained only with the

knowledge learning objective (**WKLM without MLM**), a model trained with both knowledge learning and masked language modeling with more masked words (**WKLM with** 15% **MLM**) and a BERT model trained with additional 1 million updates on English Wikipedia (**BERT + 1M MLM updates**) and no knowledge learning.

The ablation results are shown in Table 6.5. The results of WKLM without MLM validate that adding the language model objective is essential for downstream performance. We also find that masking out too many words (*i.e.*, 15% masking ratio as in the original BERT) leads to worse results. We conjecture that too many masked words outside entity mentions break parts of the context information and introduce noisy signals to knowledge learning. Results of continued BERT training show that more MLM updates are often beneficial, especially for SQuAD. However, on tasks that are more entity-centric, continued MLM training is less effective than our WKLM method. This suggests that our WKLM method could serve as an effective complementary recipe to masked language modeling when applied to entity-related NLP tasks.

Table 6.5: Ablation Studies on Masked Language Model and Masking Ratios.

Model	**SQuAD**		**TriviaQA**		**Quasar-T**	
	EM	F1	EM	F1	EM	F1
Our BERT	83.4	90.5	48.7	53.2	40.4	46.1
WKLM	84.3	**91.3**	**52.2**	**56.7**	**43.7**	**49.9**
WKLM without MLM	80.5	87.6	48.2	52.5	42.2	48.1
WKLM with 15% masking	84.1	91.0	51.0	55.3	42.9	49.0
Our BERT + 1M MLM updates	**84.4**	91.1	52.0	56.3	42.3	48.2

Chapter 7

Multi-Hop Reasoning for Complex Textual QA

7.1 Introduction

In previous chapters, most of our studies are conducted on datasets that mostly include simple information-seeking questions. To answer those questions, only a single document is required. However, despite the size of the corpus, there will always be questions that are more complex and require evidence aggregation from multiple documents. These questions are usually compositional in nature and answering them typically requires logical reasoning or sequential (multi-hop) processing (see Figure 7.1 for an example). In these scenarios, single-shot approaches to retrieval are inherently insufficient. Instead, iterative methods are needed to recursively retrieve new information at each step, conditioned on the information already at hand. Beyond further expanding the scope of existing textual open-domain QA systems, answering more complex questions usually involves *multi-hop reasoning*, which poses unique challenges for existing neural-based AI systems. With its practical and research values, multi-hop QA has

been extensively studied recently [172, 173, 174] and remains an active research area in NLP [175, 176, 177, 178, 179, 180].

The main problem in answering multi-hop open-domain questions is that the search space grows exponentially with each retrieval hop. Most recent work tackles this issue by constructing a document graph utilizing either entity linking or existing hyperlink structure in the underlying Wikipedia corpus [176, 179]. The problem then becomes finding the best path in this graph, where the search space is bounded by the number of hyperlinks in each passage. However, such methods may not generalize to new domains, where entity linking might perform poorly, or where hyperlinks might not be as abundant as in Wikipedia. Moreover, efficiency remains a challenge despite using these data-dependent pruning heuristics, with the best model [179] needing hundreds of calls to large pretrained models to produce a single answer.

In contrast, we propose to employ dense retrieval to the multi-hop setting with a simple recursive framework. Our method iteratively encodes the question and previously retrieved documents as a query vector and retrieves the next relevant documents using efficient MIPS methods. With high-quality, dense representations derived from strong pretrained encoders, our work first demonstrates that the sequence of documents that provide sufficient information to answer the multi-hop question can be accurately discovered from unstructured text, *without* the help of corpus-specific hyperlinks. When evaluated on two multi-hop benchmarks, HotpotQA [173] and a multi-evidence subset of FEVER [181], our approach improves greatly over the traditional linking-based retrieval methods. More importantly, the better retrieval results also lead to state-of-the-art downstream results on both datasets. On HotpotQA, we demonstrate a vastly improved efficiency-accuracy trade-off achieved by our system: by limiting the amount of retrieved contexts fed into downstream models, our system can match the best published result while being 10x faster.

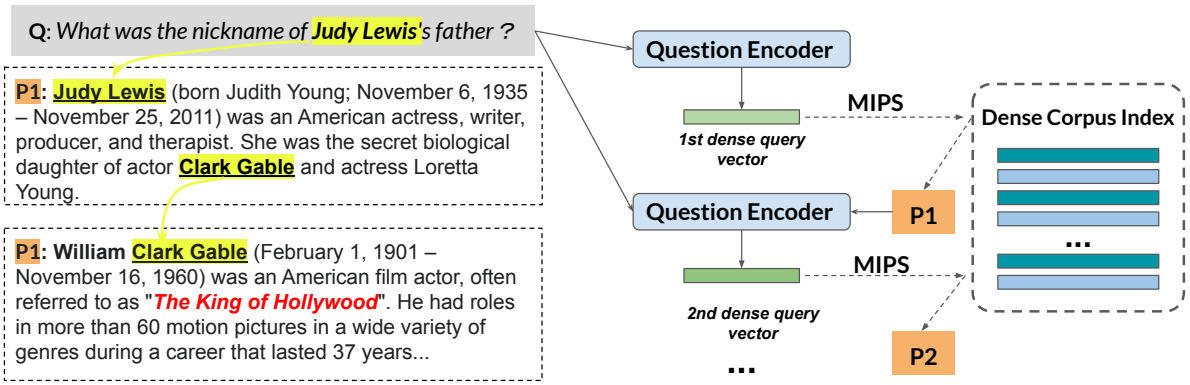


Figure 7.1: An overview of the multi-hop dense retrieval approach.

7.2 Related Work

Open-domain QA with Dense Retrieval In contrast to sparse term-index IR methods that are widely used by existing open-domain QA systems [182, 183, 168], recent systems [184, 185, 186] typically uses dense passage retrieval techniques that better capture the semantic matching beyond simple n-gram overlaps. To generate powerful dense question and passage representations, these methods either conduct large-scale pretraining with self-supervised tasks that are close to the underlying question-passage matching in retrieval, or directly use the human-labeled question-passage pairs to finetune pretrained masked language models. On single-hop information-seeking QA datasets such as NaturalQuestions [187] or WebQuestions [188], these dense methods have achieved significant improvements over traditional IR methods. Prior to these methods based on pretrained models, [189] use RNN encoder to get dense representations of questions and passages. They also consider an iterative retrieval process and reformulate the query representation based on reader model's hidden states. However, their method requires an initial round of TF-IDF/BM25 retrieval and a sophisticated RL-based training paradigm to work well. Finally, like the aforementioned methods, only single-hop datasets are considered in their

experiments. More akin to our approach, [190] use a similar recursive dense retrieval formulation for multi-hop QA. In contrast to their biattenional reformulation component, which is applied on top of the token-level representations of the query and passages, we adopt a more straightforward query reformulation strategy, by simply concatenating the original query and previous retrieval as the inputs to the query encoder. Together with stronger pretrained encoders and more effective training methods (in-batch + memory bank negative sampling vs their binary ranking loss), MDR is able to double the accuracy of their system.

Query Expansion Techniques in IR As our dense encoder augments the original question with the initial retrieved results to form the updated query representation, our work is also relevant to query expansion techniques [191, 192, 193] that are widely used in traditional IR systems. In particular, our system is similar in spirit to pseudo-relevance feedback techniques [194, 195, 196], where no additional user interaction is required at the query reformulation stage. Existing studies mainly focus on alleviating the uncertainty of the user query [197] by adding relevant terms from the first round of retrieval, where the retrieval target remains the same throughout the iterative process. In contrast, the query reformulation in our approach aims to follow the multi-hop reasoning chain and effectively retrieves different targets at each step. Furthermore, instead of explicitly selecting terms to expand the query, we simply concatenate the whole passage and rely on the pretrained encoder to choose useful information from the last retrieved passage.

Other Multi-hop QA Work Apart from HotpotQA, other multi-hop QA datasets [174, 172, 198] are mostly built from knowledge bases (KBs). Compared to questions in HotpotQA, questions in these datasets are rather synthetic and less diverse. As multi-hop relations in KBs could be mentioned together in a single text piece, these datasets are

not designed for an open-domain setting which necessitates multi-hop retrieval. Existing methods on these datasets either retrieve passages from a small passage pool pruned based on the the specific dataset [199, 200], or focus on a non-retrieval setting where a compact documents set is already given [201, 202, 203, 204]. Compared to these research, our work aims at building an efficient multi-hop retrieval model that easily scales to large real-world corpora that include millions of open-domain documents.

7.3 Recursive Dense Retrieval

7.3.1 Problem Definition

The retrieval task considered here can be described as follows (see also Figure 7.1). Given a multi-hop question q and a large text corpus $\mathcal{C}$, the retrieval module needs to retrieve a sequence of passages $\mathcal{P}_{seq} : \{p_1, p_2, ..., p_n\}$ that provide *sufficient* information for answering q. Practically, the retriever returns the k best-scoring sequence candidates, $\{\mathcal{P}^1_{seq}, \mathcal{P}^2_{seq}, ..., \mathcal{P}^k_{seq}\}$ ($k \ll |\mathcal{C}|$), with the hope that at least one of them has the desired qualities. k should be small enough for downstream modules to process in a reasonable time while maintaining adequate recall. In general, retrieval also needs to be efficient enough to handle real-world corpora containing millions of documents.

7.3.2 Approach

Model Details Based on the sequential nature of the multi-hop retrieval problem, our system solves it in an iterative fashion. We model the probability of selecting a certain passage sequence as follows:

$$P(\mathcal{P}_{seq}|q) = \prod_{t=1}^{n} P(p_t|q, p_1, ..., p_{t-1}),$$

where for $t = 1$, we only condition on the original question for retrieval. At each retrieval step, we construct a new query representation based on previous results and the retrieval is implemented as maximum inner product search over the dense representations of the whole corpus:

$$P(p_t|q, p_1, ..., p_{t-1}) = \frac{\exp\left(\langle \boldsymbol{p}_t, \boldsymbol{q}_t \rangle\right)}{\sum_{p \in \mathcal{C}} \exp\left(\langle \boldsymbol{p}, \boldsymbol{q}_t \rangle\right)}, \text{ where } \boldsymbol{q}_t = g(q, p_1, ..., p_{t-1}) \text{ and } \boldsymbol{p}_t = h(p_t).$$

Here $\langle \cdot, \cdot \rangle$ is the inner product between the query and passage vectors. $h(\cdot)$ and and $g(\cdot)$ are passage and query encoders that produce the dense representations. In order to reformulate the query representation to account for previous retrieval results at time step t, we simply concatenate the question and the retrieved passages as the inputs to $g(\cdot)$. Note that our formulation for each retrieval step is similar to existing single-hop dense retrieval methods [184, 185, 186] except that we add the query reformulation process conditioned on previous retrieval results. Additionally, instead of using a bi-encoder architecture with separately parameterized encoders for queries and passages, we use a shared RoBERTa-base [205] encoder for both $h(\cdot)$ and $g(\cdot)$. In section 7.4.1, we show this simple modification yields considerable improvements. Specifically, we apply layer normalization over the start token's representations from RoBERTa to get the final dense query/passage vectors.

Training and Inference The retriever model is trained as in [186], where each input query (which at each step consists of a question and previously retrieved passages) is paired with a positive passage and m negative passages to approximate the softmax over all passages. The positive passage is the gold annotated evidence at step t. Negative passages are a combination of passages in the current batch which correspond to other questions (*in-batch*), and *hard* negatives which are false adversarial passages. In our

experiments, we obtain hard negatives from TF-IDF retrieved passages and their linked pages in Wikipedia. We note that using hyperlinked pages as additional negatives is neither necessary nor critical for our approach. In fact we observe only a very small degradation in performance if we remove them from training (section 7.4.1). In addition to in-batch negatives, we use a memory bank ($\mathcal{M}$) mechanism [206] to further increase the number of negative examples for each question. The memory bank stores a large number of dense passage vectors. As we block the gradient back-propagation in the memory bank, its size ($|\mathcal{M}| \gg$ batch size) is less restricted by the GPU memory size. Specifically, after training to convergence with the shared encoder, we freeze a copy of the encoder as the new passage encoder and collect a bank of passage representations across multiple batches to serve as the set of negative passages. This simple extension results in further improvement in retrieval. (section 7.4.1).

For inference, we first encode the whole corpus into an index of passage vectors. Given a question, we use beam search to obtain top-k passage sequence candidates, where the candidates to beam search at each step are generated by MIPS using the query encoder at step t, and the beams are scored by the sum of inner products as suggested by the probabilistic formulation discussed above. Such inference relies only on the dense passage index and the query representations, and does not need explicit graph construction using hyperlinks or entity linking. The top-k sequences will then be fed into task-specific downstream modules to produce the desired outputs.

7.4 Experiments

Datasets Our experiments focus on two datasets: *HotpotQA* and *Multi-evidence FEVER*. HotpotQA [173] includes 113k multi-hop questions. Unlike other multi-hop QA datasets [198, 172, 174], where the information sources of the answers are knowledge bases, HotpotQA

uses documents in Wikipedia. Thus, its questions are not restricted by the fixed KB schema and can cover more diverse topics. Each question in HotpotQA is also provided with ground truth support passages, which enables us to evaluate the intermediate retrieval performance. Multi-evidence FEVER includes 20k claims from the FEVER [181] fact verification dataset, where the claims can only be verified using multiple documents. We use this dataset to validate the general applicability of our method.

Implementation Details All the experiments are conducted on a machine with 8 32GB V100 GPUs. Our code is based on Huggingface Transformers [207]. Our best retrieval results are predicted using the exact inner product search index (IndexFlatIP) in FAISS [208]. Both datasets assume 2 hops, so we fix $n = 2$ for all experiments. Since HotpotQA does not provide the order of the passage sequences, as a heuristic, we consider the passage that includes the answer span as the final passage. [1] In section 7.4.1, we show that the order of the passages is important for effective retriever training.

7.4.1 Experiments: Retrieval

We evaluate our multi-hop dense retriever (MDR) in two different use cases: *direct* and *reranking*, where the former outputs the top-k results directly using the retriever scores and the latter applies a task-specific reranking model to the initial results from MDR.

Direct

We first compare MDR with several efficient retrieval methods that can directly find the top-k passage sequences from a large corpus, including TF-IDF, TF-IDF + Linked,

[1] If the answer span is in both, the one that has its title mentioned in the other passage is treated as the second.

Table 7.1: Retrieval performance in recall at k retrieved passages and precision/recall/F_1.

Method	HotpotQA			FEVER		
	R@2	R@10	R@20	Precision	Recall	F_1
TF-IDF	10.3	29.1	36.8	14.9	28.2	19.5
TF-IDF + Linked	17.3	50.0	62.7	18.6	35.8	24.5
DrKIT	38.3	67.2	71.0	-	-	-
Entity Linking	-	-	-	30.6	53.8	39.0
MDR	**65.9**	**77.5**	**80.2**	**45.7**	**69.1**	**55.0**

DrKIT and Entity Linking. **TF-IDF** is the standard term-matching baseline, while **TF-IDF + Linked** is a straightforward extension that also extracts the hyperlinked passages from TF-IDF passages, and then reranks both TF-IDF and hyperlinked passages with BM25 [2] scores. **DrKIT** [200] is a recently proposed dense retrieval approach, which builds a entity-level (mentions of entities) dense index for retrieval. It relies on hyperlinks to extract entity mentions and prunes the search space with a binary mask that restricts the next hop to using hyperlinked entities. On FEVER, we additionally consider an entity linking baseline [209] that is commonly used in existing fact verification pipelines. This baseline first uses a constituency parser to extract potential entity mentions in the fact claim and then uses the MediaWiki API to search documents with titles that match the mentions.

Table 7.1 shows the performance of different retrieval methods. On HotpotQA the metric is recall at the top k paragraphs[3], while on FEVER the metrics are precision, recall and F_1 in order to be consistent with previous results. On both datasets, MDR substantially outperforms all baselines.

[2] https://pypi.org/project/rank-bm25

[3] As the sequence length is 2 for HotpotQA, we pick the top $k/2$ sequences predicted by MDR.

Reranking

Reranking documents returned by efficient retrieval methods with a more sophisticated model is a common strategy for improving retrieval quality. For instance, state-of-the-art multi-hop QA systems usually augment traditional IR techniques with large pretrained language models to select a more compact but precise passage set. On HotpotQA, we test the effectiveness of MDR after a simple cross-attention reranking: each of the top k passage sequences from MDR is first prepended with the original question and then fed into a pretrained Transformer encoder, i.e., ELECTRA-large [210], that predicts relevant scores. We train this reranking model with a binary cross-entropy loss, with the target being whether the passage sequence cover both groundtruth passages. We empirically compare our approach with two other existing reranking-based retrieval methods: **Semantic Retrieval** [176] uses BERT at both passage-level and sentence-level to select context from the initial TF-IDF and hyperlinked passages; **Graph Recurrent Retriever** [179] learns to recursively select the best passage sequence on top of a hyperlinked passage graph, where each passage node is encoded with BERT.

Table 7.2: HotpotQA reranked retrieval results (input passages for final answer prediction).

Method	SP EM	Ans Recall
Semantic Retrieval	63.9	77.9
Graph Rec Retriever	75.7	87.5
MDR (direct)	65.9	75.4
MDR (reranking)	**81.2**	**88.2**

Table 7.3: Retriever Model Ablation on HotpotQA retrieval. *Single-hop* here is equivalent to the DPR method [186].

Retriever variants	R@2	R@10	R@20
Full Retrieval Model	65.9	77.5	80.2
- w/o linked negatives	64.6	76.8	79.6
- w/o memory bank	63.7	74.2	77.2
- w/o shared encoder	59.9	70.6	73.1
- w/o order	17.6	55.6	62.3
Single-hop	25.2	45.4	52.1

Table 7.2 shows the reranking results. Following [179], we use *Answer Recall* and *Support Passage Exact Match (SP EM)* [4] as the evaluation metrics. Even without reranking,

[4]Whether the final predicted sequence covers both gold passages.

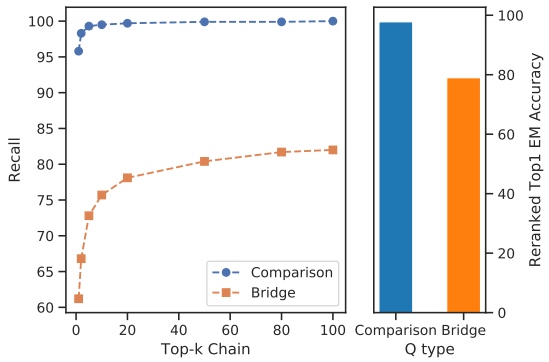


Figure 7.2: The retrieval performance gap between comparison and bridge questions. Left: recall of groundtruth passage sequences without reranking. Right: Top-1 chain exact match after reranking.

MDR is already better than Semantic Retrieval, which requires around 50 BERT encoding (where each encoding involves cross-attention over a concatenated question-passage pair). After we rerank the top-100 sequences from the dense retriever, our passage recall is better than the state-of-the-art Graph Recurrent Retriever, which uses BERT to process more than 500 passages. We do not compare the reranked results on FEVER, as most FEVER systems directly use BERT encoder to select the top evidence *sentences* from the retrieved documents, instead of the reranking the documents.

Analysis

To understand the strengths and weaknesses of MDR, we conduct further analysis on HotpotQA dev.

Retrieval Error Analysis HotpotQA contains two question categories: *bridge* questions in which an intermediate entity is missing and needs to be retrieved before inferring the answer; and *comparison* questions where two entities are mentioned simultaneously and compared in some way. In Figure 7.2, we show the retrieval performance of both question types. The case of *comparison* questions proves easier, since both entities needed for

retrieval are present in the question. This case appears almost solved, confirming recent work demonstrating that dense retrieval is very effective at entity linking [211].

For the case of *bridge* questions, we manually inspect 50 randomly sampled erroneous examples after reranking. We find that in half of these cases, our retrieval model predicts an alternative passage sequence that is also valid. This gives an estimated top-1 passage sequence accuracy of about 90%. Other remaining errors are due to the dense method's inability to capture the exact n-gram match between the question and passages. This is a known issue [184, 186] of dense retrieval methods when dealing with questions that have high lexical overlap with the passages. To this end, a hybrid multi-hop retrieval method with both term and dense index might be used to further improve the performance on *bridge* questions.

Retriever Ablation Study In Table 7.3, we examine our model with different variations on HotpotQA to show the effectiveness of each proposed component. We see that further training with a memory bank results in modest gains, while using a shared encoder is crucial for the best performance. Respecting the ordering of passages in two hops is essential - training in an order-agnostic manner hardly works at all, and underperforms even the single-hop baseline. Finally, not using hyperlinked paragraphs from TF-IDF passages as additional negatives has only a minor impact on performance.

Question Decomposition for Retrieval As multi-hop questions have more complex structures than simple questions, recent studies [177, 180] propose to use explicit question decomposition to simplify the problem. [212] shows that with TF-IDF, using decomposed questions improves the retrieval results. We investigate whether the conclusion still holds with stronger dense retrieval methods. We use the human-annotated question decomposition from the QDMR dataset [212] for analysis. For a question like `Q:Mick Carter is`

Table 7.4: Comparison with decomposed dense retrieval which uses oracle question decomposition (test on 100 bridge questions). See text for details about the decomposed settings.

Method	R@2	R@10	R@20
MDR	54.9	63.7	70.6
Decomp (SubQ1;SubQ2)	50.0	64.7	67.6
Decomp (Q;SubQ2)	51.0	64.7	68.6

`the landlord of a public house located at what address?`, QDMR provides two subquestions, `SubQ1: What is the public house that Mick Carter is the landlord of?` and `SubQ2: What is the address that #1 is located at?`. We sample 100 bridge questions and replace `#1` in `SubQ2` with the correct answer (The Queen Victoria) to `SubQ1`. Note that this gives advantages to the decomposed method as we ignore any intermediate errors. We estimate the performance of potential decomposed methods with the state-of-the-art single-hop dense retrieval model [186].

As shown in Table 7.4, we did not observe any strong improvements from explicit question decomposition, which is contrary to the findings by [212] when using term-based IR methods. Moreover, as shown in the third row of the table, when the 1st hop of the decomposed retrieval (i.e., `SubQ1`) is replaced with the original question, no performance degradation is observed. This suggests that strong pretrained encoders can effectively learn to select necessary information from the multi-hop question at each retrieval step. Regarding the performance drop when using explicit compositions, we hypothesize that it is because some information in one decomposed subquestion could be useful for the other retrieval hop. While this could potentially be addressed by a different style of decomposition, our analysis suggests that decomposition approaches might be sub-optimal in the context of dense retrieval with strong pretrained encoders.

7.4.2 Experiments: HotpotQA

We evaluate how the better retrieval results of MDR improve multi-hop question answering in this section. As our retriever system is agnostic to downstream models, we test two categories of answer prediction architectures: the *extractive* span prediction models based on pretrained masked language models, such as BERT [18] and ELECTRA [210], and the retrieval-augmented *generative* reader models [213, 214], which are based on pretrained sequence-to-sequence (seq2seq) models such as BART [215] and T5 [216]. Note that compared to more complicated graph reasoning models [217, 178], these two classes of models do not rely on hyperlinks and can be applied to any text.

Extractive reader models learn to predict an answer span from the concatenation of the question and passage sequence ($[q, p_1, ..., p_n]$). On top of the token representations produced by pretrained models, we add two prediction heads to predict the start and end position of the answer span.[5] To predict the supporting sentences, we add another prediction head and predict a binary label at each sentence start. For simplicity, the same encoder is also responsible for reranking the top k passage sequences. The reranking detail has been discussed in section 7.4.1. Our best reader model is based on ELECTRA [210], which has achieved the best single-model performance on the standard SQuAD [218] benchmark. Additionally, we also report the performance of BERT-large with whole word masking (BERT-wwm) to fairly compare with [179].

Generative models, such as RAG [213] and FiD [214], are based on pretrained seq2seq models. These methods finetune pretrained models with the concatenated questions and retrieved documents as inputs, and answer tokens as outputs. This generative paradigm has shown state-of-the-art performance on single-hop open-domain QA tasks. Specifically, FiD first uses the T5 encoder to process each retrieved passage sequence

[5]To account for yes/no questions, we prepend *yes* and *no* tokens to the context.

independently and then uses the decoder to perform attention over the representations of all input tokens while generating answers. RAG is built on the smaller BART model. Instead of only tuning the seq2seq model, it also jointly train the question encoder of the dense retriever. We modified it to allow multi-hop retrieval.

Retrieval-Free Approaches

Before evaluating the aforementioned retrieval-dependent models, we first investigate whether it is necessary to drop the the retrieval component and relies on the large capacity of pretrained models to answer questions. Inspired by a recent work [219] that trains the T5 seq2seq model to directly decode answers from questions (*retrieval-free*), we conduct similar experiments on HotpotQA using BART [215]. We compare it with existing state-of-the-art retrieval-based approaches. As shown in Figure 7.3, the performance gap between retrieval-based methods and retrieval-free methods on multi-hop QA is much larger than the gap in the case of simple single-hop questions.

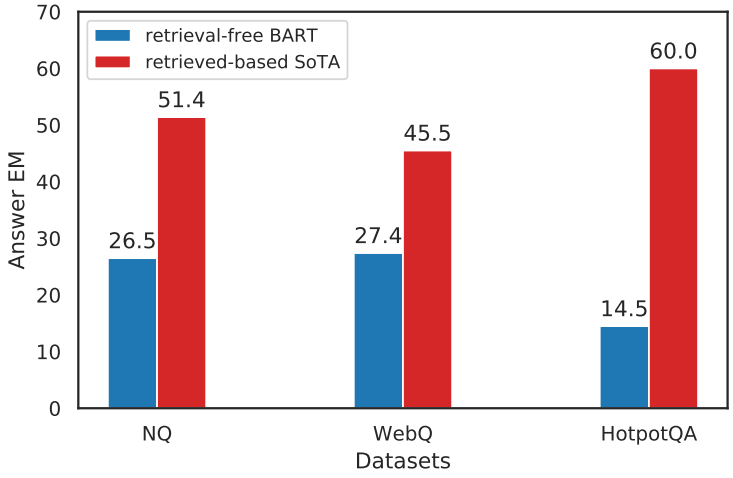


Figure 7.3: Performance gap between retrieval-free and retrieval-based methods on different QA datasets.

Results

Table 7.5: HotpotQA-fullwiki test results.

Methods	Answer		Support		Joint	
	EM	F1	EM	F1	EM	F1
GoldEn Retriever [175]	37.9	48.6	30.7	64,2	18.9	39.1
Semantic Retrieval [176]	46.5	58.8	39.9	71.5	26.6	49.2
Transformer-XH [178]	51.6	64.1	40.9	71.4	26.1	51.3
HGN [217]	56.7	69.2	50.0	76.4	35.6	59.9
DrKIT [200]	42.1	51.7	37.1	59.8	24.7	42.9
Graph Recurrent Retriever [179]	60.0	73.0	49.1	76.4	35.4	61.2
MDR (ELECTRA Reader)	**62.3**	**75.3**	**57.5**	**80.9**	**41.8**	**66.6**

Comparison with Existing Systems Table 7.5 compares the HotpotQA test performance of our best ELECTRA reader with recently published systems, using the numbers from the official leaderboard, which measure answer and supporting sentence exact match (EM)/F1 and joint EM/F1. Among these methods, only GoldEn Retriever [175] does not exploit hyperlinks. In particular, Graph Recurrent Retriever trains a graph traversal model for chain retrieval; TransformerXH [178] and HGN [217] explicitly encode the hyperlink graph structure within their answer prediction models. In fact, this particular inductive bias provides a perhaps unreasonably strong advantage in the specific context of HotpotQA, which by construction guarantees ground-truth passage sequences to follow hyperlinks. Despite not using such prior knowledge, our model outperforms all previous systems by large margins, especially on supporting fact prediction, which benefits more directly from better retrieval.

Table 7.6: Reader comparison on HotpotQA dev set.

	Model	Top k	EM	F1
Extractive	ELECTRA	Top 50	61.7	74.3
	ELECTRA	Top 250	63.4	76.2
	BERT-wwm	Top 250	61.5	74.7
Generative	Multi-hop RAG	Top 4*4	51.2	63.9
	FiD	Top 50	61.7	73.1

Reader Model Variants Results for reader model variants are shown in Table 7.6.[6] First, we see that the BERT-wwm reader is 1-2% worse than the ELECTRA reader when using enough passages. However, it still outperforms the results in [179] which also uses BERT-wwm for answer prediction. While RAG and FiD have shown strong improvements over extractive models on single-hop datasets such as NaturalQuestions [187], they do not show an advantage in the multi-hop case. Despite having twice as many parameters as ELECTRA, FiD fails to outperform it using the same amount of context (top 50). In contrast, on NaturalQuestions, FiD is 4 points better than a similar extractive reader when using the top 100 passages in both.[7] We hypothesize that the improved performance on single-hop questions is due to the ability of larger pretrained models to more effectively memorize single-hop knowledge about real-world entities.[8] Compared to multi-hop questions that involve multiple relations and missing entities, simple questions usually only ask about a certain property of an entity. It is likely that such simple entity-centric information is explicitly mentioned by a single text piece in the pretrain-

[6]For the compute-heavy generative models, we feed in as many passages as possible without running into memory issues (Muli-hop RAG takes top 4 passages from hop1, and for each of those, takes another top 4 from hop2. They are not necessarily the same as the top 16 passages sequences.). As extractive models encode each passage sequence separately, we can use arbitrary number of input sequences. However, the performance mostly plateaus as we use over 200 input sequences.

[7]We implemented NQ extractive readers with both RoBERTa-large and ELECTRA-large, and RoBERTa-large yielded a better answer EM of 47.3, which is much lower than the 51.4 answer EM achieved by FiD.

[8]As shown by [219], a large pretrained seq2seq model can be finetuned to directly decode answers with questions as the only inputs. However, as we showed earlier, this retrieval-free approach performs poorly on multi-hop questions.

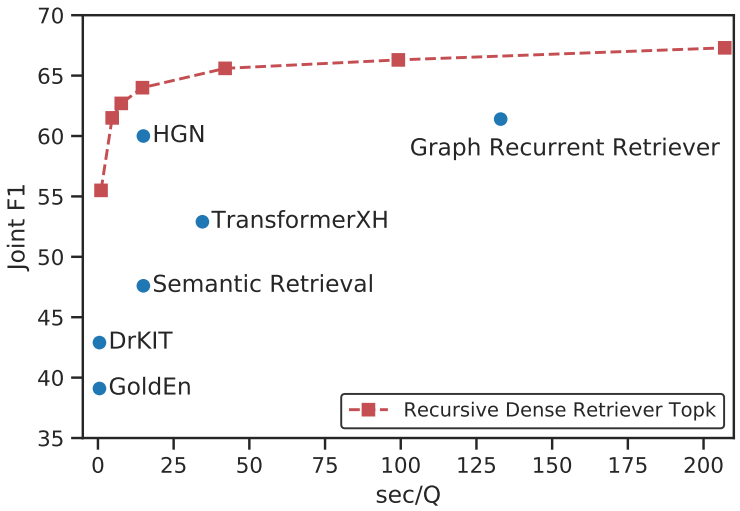


Figure 7.4: Efficiency-performance trade-off comparison with published HotpotQA systems. The curve is plotted with different number of top k (k=1,5,10,20,50,100,200) passage sequences we feed into the reader model. seq/Q denotes the time required for each query.

ing corpus, while the evidence for multihop questions is typically dispersed, making the complete reasoning chain nontrivial to memorize.

Inference Efficiency To compare with existing multi-hop QA systems in terms of efficiency, we follow [200] and measure the inference time with 16 CPU cores and batch size 1. We implement our system with a fast approximate nearest neighbor search method, *i.e.*, HNSW [220], which achieves nearly the same performance as exact search. With an in-memory index, we observe that the retrieval time is negligible compared to the forward pass of large pretrained models. Similarly, for systems that use term-based indices, the BERT calls for passage reranking cause the main efficiency bottleneck. Thus, for systems that do not release the end-to-end code, we estimate the running time based on the number of BERT cross-attention forward passes (the same estimation strategy used by [200]), and ignore the overhead caused by additional processing such as TF-IDF or linking graph construction. As shown in Figure 7.4, our method is about 10 times faster than current

Table 7.7: Multi-Evidence FEVER Fact Verification Results. **Loose-Multi** represents the subset that requires multiple evidence *sentences*. **Strict-Multi** is a subset of **Loose-Multi** that require multiple evidence sentences from different *documents*.

Method	Loose-Multi (1,960)		Strict-Multi (1,059)	
	LA	FEVER	LA	FEVER
GEAR	66.4	38.0	-	-
GAT	66.1	38.2	-	-
KGAT with ESIM rerank	65.9	39.2	51.5	7.7
KGAT with BERT rerank	65.9	40.1	51.0	6.2
Ours + KGAT with BERT rerank	**77.9**	**42.0**	**72.1**	**16.2**

state-of-the-art systems while achieving a similar level of performance. Compared to two efficient systems (DrKIT and GoldEn), we achieve over 10 points improvement while only using the top-1 retrieval result for answer and supporting sentence prediction.

7.4.3 Experiments: Multi-Evidence FEVER

For FEVER claim verification, we reuse the best open-sourced verification system, *i.e.*, KGAT [221], to show the benefit of our retrieval approach over existing retrieval methods. We report the results in verification *label accuracy* (LA) and the *FEVER score*[9] in Table 7.7, where the numbers of competitive baselines, GEAR [222], graph attention network (GAT) [223] and variants of KGAT are from the KGAT [221] paper. All these baselines use entity linking for document retrieval, then rerank the sentences of the retrieved documents, and finally use different graph attention mechanisms over the fully-connected sentence graph to predict verification labels. Since some instances in the multi-evidence subset used by previous studies only needs multiple evidence *sentences* from the same document, we additionally test on a strict multi-hop subset with instances that need multiple *documents*. As shown by the results, even without finetun-

[9]FEVER scores takes into account both support sentence accuracy and label accuracy, similar as the joint metrics in HotpotQA.

ing the downstream modules, simply replacing the retrieval component with MDR leads to significant improvements, especially on the strict multi-evidence subset.

7.4.4 Experiments: A Unified QA Retrieval System

In practice, when a fixed text corpus is given for open-domain systems, we do not know beforehand whether the incoming questions require single or multiple text evidence. Thus, it is essential to build a unified system that adaptively retrieves for multiple hops. Due to the simplicity of the approach, our method can easily be extended in the unified setup. To the best of our knowledge, only [179] test the same retrieval method on both single and multi-hop questions but with separate trained models. Here we take a further step and explore the possibility of using a single retrieval model for both types of questions.

To enable adaptive retrieval, we add a binary prediction head on top of the question encoder. Once the retriever finishes the 1-hop retrieval, it encodes concatenation of q and p_1 and predicts whether to stop retrieval using the final hidden state of the first token. We construct this unified setting with NaturalQuestions-Open [184] (NQ) as single-hop and HotpotQA as multi-hop. As the two datasets use different corpora, we merge the two[10] for easy comparison. As baselines, we use the retrieval models trained only on the respective dataset. For HotpotQA, the baseline is the best multi-hop retrieval model discussed in the main text. For NQ, we follow the training method in DPR [186], but with a shared question and passage encoder, which achieves stronger results. As the NQ corpus includes multiple passages of the same document and the HotpotQA corpus only uses the introduction passage, we are not able to compute the strict title-based support passage recall for HotpotQA as in section 7.4.2. Thus, we only evaluate answer recall. Results are in Table 7.8. In contrast to existing studies that train different models for

[10]The Wikipedia corpus of NQ is taken from DPR [186].

each dataset, we show that a unified dense retrieval model can maintain competitive performance on both, despite the vastly different nature of both datasets. Note that the information-seeking questions in NQ is usually noisier and more ambiguous, while HotpotQA questions are more complicated and contains more lexical overlaps with the evidence passages. Specifically, for NQ, the unified retrieval model achieves very similar performance as the single-dataset DPR model, while the performance on HotpotQA decreases more. We conjecture that this is because the information-seeking questions in NQ cover more diverse patterns, and the added HotpotQA training questions do not cause a dramatic distribution shift from the NQ test data. We leave the development of a more general retrieval system that handles different styles of questions to future work.

Table 7.8: Comparing the unified retrieval model with models specifically trained for each task. We test the retrieval performance with a single merged corpus. For easy comparison, all three models are based on BERT-base encoder which we find achieves stronger performance than RoBERTa-base on NQ. AR@K denotes answer recall at top-K retrieved passage sequences.

Model	NQ		HotpotQA	
	AR@20	AR@100	AR@20	AR@100
single-hop only	80.7	87.3	-	-
multi-hop only	-	-	83.4	89.4
unified	79.5	86.1	78.1	83.0

Chapter 8

Conclusions

In this dissertation, we have covered various techniques to build state-of-the-art neural QA systems, either using structured knowledge source or unstructured text.

In Chapter 1, we first briefly introduced the history of NLP and gave the motivations to study the question answering problem. We then described the two prominent paradigm to build state-of-the-art QA systems. These two paradigms have their own pros and cons. The structured KB approach reduces the ambiguity of the knowledge source and the database query languages also make it easier to efficient retrieve the desired information. In contrast, using text corpora to answer questions involves more complicated modeling. However, the textual knowledge source is much easier and cheaper to build and maintain. More important, unstructured text data offers a wider knowledge scope and has the potential to answer more types of questions.

In the first part of the dissertation, we discussed our efforts to automatically enrich KBs. We presented a RL based approach which is both accurate and interpretable (Chapter 2). The proposed RL reasoning agent can effectively learn from weak supervision. In Chapter 3, we also presented a metric-learning based approach that can complete relation types that are long-tail and do not have many training instances. This few-shot learning

method is built on pretrained KB embeddings and graph neural networks. It can leverage the local neighbor graph to provide more information in the low-resource scenarios. In Chapter 4, we took a different angle and tried to develop a hybrid QA model that can use knowledge source from both domains. With graph networks and bidirectionally attention matching, this approach can improves the coverage of KBQA and also the machine reading over text. It is worth noting that the three proposed approaches are orthogonal to each other can be combined to improve an KB-based QA system.

In Part II, we began to look at textual based QA systems. Due to large size and unstructured nature of text corpora, an efficient text retrieval module is usually essential in such systems. In Chapter 5, we discussed the limitation of traditional information retrieval systems. Compared to the sparse TF-IDF style text representations, dense representations learned from neural networks can capture deep semantic matching between questions and documents. However, in practical weak-supervision scenarios, i.e., only the QA pairs are available for training, existing dense retrieval methods required expensive pretraining. We proposed a simple and effective clustering-based approach to improve the efficiency of pretraining. Combined with synthetic data generated from pretrained seq2seq models, we achieved improvements both in accuracy and efficiency. In Chapter 6, we took a closer looking at the answer extraction module in textual QA systems. While the masked language modeling objective used in various pretrained models have achieved substantial improvements over previous models, it does not emphasize the knowledge skills required in question answering. We investigated whether enforcing an explicit knowledge-aware training objective could further improve QA. Finally, in Chapter 7, we studied how to efficiently retrieve and aggregate multiple evidence to answer complex questions from text. We proposed a simple and general recursive dense retrieval method which does rely on any corpus-specific structures. We explored both generative and extractive models for this kind of multi-hop QA. We achieve state-of-the-art perfor-

mance on a popular multi-hop QA benchmark while maintaining superior efficiency than many existing systems.

Overall, the outline of this dissertations aligns with the recent trend we have seen in the research community: more efforts have been put into textual QA systems compared the previously dominant KBQA paradigm. However, this does not indicate that text-based QA systems can already replace the KBQA components in various applications. For instance, with the help of graph query languages, it is still more straightforward to use KBs to answer questions that require aggregation operations, such as argmax, count, etc. Also, in certain domains where the precision is more important than recall, constructing a KB that covers the most probable questions might still be a better option. At least in the short-term future, we still expect to see more research on combining the multi-modal resources to build better hybrid QA systems. We also believe that answering questions from text is a more promising paradigm and will attract more research efforts in the long term. Not only because textual QA brings more fundamental NLP challenges but also its simpler implementation pipelines. To further improve textual QA systems, there are two challenges which we believe to be essential. First, aligned with our work described in Chapter 7, we need modules to handle questions that involve arbitrary number of text documents and aggregate the evidence in an efficient and also interpretable way. Second, we observe that existing pretrained models could only handle a limited window of context and we need scalable methods that can capture the long-term dependencies. These methods will allow us to further improve the scope of QA systems in terms of those questions that require a deeper understanding and reasoning. We hope this dissertations could provide insights from both QA paradigms and inspire future research towards building more intelligent tools that make it easier for users to acquire any kind of knowledge from the Web.

Bibliography

[1] S. Russell and P. Norvig, *Artificial intelligence: a modern approach*, .

[2] 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.

[3] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, *Dbpedia: A nucleus for a web of open data*, in *The semantic web*, pp. 722–735. Springer, 2007.

[4] D. Vrandečić and M. Krötzsch, *Wikidata: a free collaborative knowledge base*, .

[5] Q. Cai and A. Yates, *Large-scale semantic parsing via schema matching and lexicon extension*, in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Sofia, Bulgaria), pp. 423–433, Association for Computational Linguistics, Aug., 2013.

[6] T. Kwiatkowski, E. Choi, Y. Artzi, and L. Zettlemoyer, *Scaling semantic parsers with on-the-fly ontology matching*, in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, (Seattle, Washington, USA), pp. 1545–1556, Association for Computational Linguistics, Oct., 2013.

[7] 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*, (Seattle, Washington, USA), pp. 1533–1544, Association for Computational Linguistics, Oct., 2013.

[8] P. Liang, *Lambda dependency-based compositional semantics*, *arXiv preprint arXiv:1309.4408* (2013).

[9] J. Berant and P. Liang, *Semantic parsing via paraphrasing*, in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1415–1425, 2014.

[10] 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)*, (Beijing, China), pp. 1321–1331, Association for Computational Linguistics, July, 2015.

[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)*, (Vancouver, Canada), pp. 23–33, Association for Computational Linguistics, July, 2017.

[12] 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)*, (Doha, Qatar), pp. 615–620, Association for Computational Linguistics, Oct., 2014.

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

[14] K. M. Hermann, T. Kočiskỳ, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, *Teaching machines to read and comprehend*, *arXiv preprint arXiv:1506.03340* (2015).

[15] D. Chen, J. Bolton, and C. D. Manning, *A thorough examination of the cnn/daily mail reading comprehension task*, *arXiv preprint arXiv:1606.02858* (2016).

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

[17] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi, *Bidirectional attention flow for machine comprehension*, *arXiv preprint arXiv:1611.01603* (2016).

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

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, *Attention is all you need*, *arXiv preprint arXiv:1706.03762* (2017).

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *Imagenet classification with deep convolutional neural networks*, in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[21] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, *et. al.*, *Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups*, *IEEE Signal Processing Magazine* **29** (2012), no. 6 82–97.

[22] Y. Kim, *Convolutional neural networks for sentence classification*, *arXiv preprint arXiv:1408.5882* (2014).

[23] N. Lao, J. Zhu, X. Liu, Y. Liu, and W. W. Cohen, *Efficient relational learning with hidden variable detection.*, in *NIPS*, pp. 1234–1242, 2010.

[24] N. Lao, T. Mitchell, and W. W. Cohen, *Random walk inference and learning in a large scale knowledge base*, in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 529–539, Association for Computational Linguistics, 2011.

[25] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, *Translating embeddings for modeling multi-relational data*, in *Advances in neural information processing systems*, pp. 2787–2795, 2013.

[26] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et. al.*, *Human-level control through deep reinforcement learning*, *Nature* **518** (2015), no. 7540 529–533.

[27] 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 *AAAI*, 2010.

[28] N. Lao, T. M. Mitchell, and W. W. Cohen, *Random walk inference and learning in a large scale knowledge base*, in *EMNLP*, pp. 529–539, ACL, 2011.

[29] M. Gardner, P. P. Talukdar, B. Kisiel, and T. M. Mitchell, *Improving learning and inference in a large knowledge-base using latent syntactic cues.*, in *EMNLP*, pp. 833–838, 2013.

[30] M. Gardner, P. P. Talukdar, J. Krishnamurthy, and T. Mitchell, *Incorporating vector space similarity in random walk inference over knowledge bases*, .

[31] W. Y. Wang and W. W. Cohen, *Joint information extraction and reasoning: A scalable statistical relational learning approach*, in *ACL*, 2015.

[32] K. Toutanova, D. Chen, P. Pantel, H. Poon, P. Choudhury, and M. Gamon, *Representing text for joint embedding of text and knowledge bases*, in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1499–1509, 2015.

[33] K. Guu, J. Miller, and P. Liang, *Traversing knowledge graphs in vector space*, in *EMNLP*, 2015.

[34] D. Zeng, K. Liu, S. Lai, G. Zhou, J. Zhao, *et. al.*, *Relation classification via convolutional deep neural network.*, in *COLING*, pp. 2335–2344, 2014.

[35] A. Neelakantan, B. Roth, and A. McCallum, *Compositional vector space models for knowledge base completion*, *arXiv preprint arXiv:1504.06662* (2015).

[36] R. Das, A. Neelakantan, D. Belanger, and A. McCallum, *Chains of reasoning over entities, relations, and text using recurrent neural networks*, *EACL* (2017).

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

[38] J. Johnson, B. Hariharan, L. van der Maaten, J. Hoffman, L. Fei-Fei, C. L. Zitnick, and R. Girshick, *Inferring and executing programs for visual reasoning*, *arXiv preprint arXiv:1705.03633* (2017).

[39] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, *Playing atari with deep reinforcement learning*, *arXiv preprint arXiv:1312.5602* (2013).

[40] 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 Conference on Artificial Intelligence (AAAI 2010)*, 2010.

[41] Z. Wang, J. Zhang, J. Feng, and Z. Chen, *Knowledge graph embedding by translating on hyperplanes.*, in *AAAI*, pp. 1112–1119, Citeseer, 2014.

[42] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et. al.*, *Mastering the game of go with deep neural networks and tree search*, *Nature* **529** (2016), no. 7587 484–489.

[43] R. J. Williams, *Simple statistical gradient-following algorithms for connectionist reinforcement learning*, *Machine learning* **8** (1992), no. 3-4 229–256.

[44] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, in *3rd International Conference on Learning Representations, ICLR 2015* (Y. Bengio and Y. LeCun, eds.), (San Diego, California, USA), 2015.

[45] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, *Learning entity and relation embeddings for knowledge graph completion.*, in *AAAI*, vol. 15, pp. 2181–2187, 2015.

[46] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, *Knowledge graph embedding via dynamic mapping matrix.*, in *ACL (1)*, pp. 687–696, 2015.

[47] M. Nickel, V. Tresp, and H.-P. Kriegel, *A three-way model for collective learning on multi-relational data*, in *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 809–816, 2011.

[48] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng, *Embedding entities and relations for learning and inference in knowledge bases*, *arXiv preprint arXiv:1412.6575* (2014).

[49] 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, 2016.

[50] N. Lao and W. W. Cohen, *Relational retrieval using a combination of path-constrained random walks*, *Machine learning* **81** (2010), no. 1 53–67.

[51] 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*, *arXiv preprint arXiv:1711.05851* (2017).

[52] W. Chen, W. Xiong, X. Yan, and W. Wang, *Variational knowledge graph reasoning*, *arXiv preprint arXiv:1803.06581* (2018).

[53] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, *Convolutional 2d knowledge graph embeddings*, *arXiv preprint arXiv:1707.01476* (2017).

[54] B. Shi and T. Weninger, *Open-world knowledge graph completion*, *arXiv preprint arXiv:1711.03438* (2017).

[55] R. Xie, Z. Liu, J. Jia, H. Luan, and M. Sun, *Representation learning of knowledge graphs with entity descriptions.*, in *AAAI*, pp. 2659–2665, 2016.

[56] G. Koch, *Siamese neural networks for one-shot image recognition.* PhD thesis, University of Toronto, 2015.

[57] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, *et. al.*, *Matching networks for one shot learning*, in *Advances in Neural Information Processing Systems*, pp. 3630–3638, 2016.

[58] J. Snell, K. Swersky, and R. S. Zemel, *Prototypical networks for few-shot learning*, *arXiv preprint arXiv:1703.05175* (2017).

[59] M. Yu, X. Guo, J. Yi, S. Chang, S. Potdar, Y. Cheng, G. Tesauro, H. Wang, and B. Zhou, *Diverse few-shot text classification with multiple metrics*, *arXiv preprint arXiv:1805.07513* (2018).

[60] S. Ravi and H. Larochelle, *Optimization as a model for few-shot learning*, in *International Conference on Learning Representations*, 2017.

[61] T. Munkhdalai and H. Yu, *Meta networks*, *arXiv preprint arXiv:1703.00837* (2017).

[62] C. Finn, P. Abbeel, and S. Levine, *Model-agnostic meta-learning for fast adaptation of deep networks*, *arXiv preprint arXiv:1703.03400* (2017).

[63] Z. Li, F. Zhou, F. Chen, and H. Li, *Meta-sgd: Learning to learn quickly for few shot learning*, *arXiv preprint arXiv:1707.09835* (2017).

[64] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, *Human-level concept learning through probabilistic program induction*, *Science* **350** (2015), no. 6266 1332–1338.

[65] Y. Duan, M. Andrychowicz, B. Stadie, O. J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba, *One-shot imitation learning*, in *Advances in neural information processing systems*, pp. 1087–1098, 2017.

[66] Y. Lin, Z. Liu, H. Luan, M. Sun, S. Rao, and S. Liu, *Modeling relation paths for representation learning of knowledge bases*, *arXiv preprint arXiv:1506.00379* (2015).

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

[68] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, *Deep sets*, in *Advances in Neural Information Processing Systems*, pp. 3394–3404, 2017.

[69] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, *Dropout: A simple way to prevent neural networks from overfitting*, *The Journal of Machine Learning Research* **15** (2014), no. 1 1929–1958.

[70] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. v. d. Berg, I. Titov, and M. Welling, *Modeling relational data with graph convolutional networks*, *arXiv preprint arXiv:1703.06103* (2017).

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

[72] L. Song, Y. Zhang, Z. Wang, and D. Gildea, *A graph-to-sequence model for amr-to-text generation*, *arXiv preprint arXiv:1805.02473* (2018).

[73] S. Hochreiter and J. Schmidhuber, *Long short-term memory*, *Neural computation* **9** (1997), no. 8 1735–1780.

[74] O. Vinyals, S. Bengio, and M. Kudlur, *Order matters: Sequence to sequence for sets*, *arXiv preprint arXiv:1511.06391* (2015).

[75] F. Mahdisoltani, J. Biega, and F. M. Suchanek, *Yago3: A knowledge base from multilingual wikipedias*, in *CIDR*, 2013.

[76] T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, B. Yang, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, *et. al.*, *Never-ending learning*, *Communications of the ACM* **61** (2018), no. 5 103–115.

[77] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, *Automatic differentiation in pytorch*, .

[78] C. Xiong, V. Zhong, and R. Socher, *DCN+: mixed objective and deep residual coattention for question answering*, *CoRR* **abs/1711.00106** (2017) [arXiv:1711.0010].

[79] A. W. Yu, D. Dohan, M. Luong, R. Zhao, K. Chen, M. Norouzi, and Q. V. Le, *Qanet: Combining local convolution with global self-attention for reading comprehension*, *CoRR* **abs/1804.09541** (2018) [arXiv:1804.0954].

[80] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, *Graph attention networks*, *arXiv preprint arXiv:1710.10903* (2017).

[81] T. H. Haveliwala, *Topic-sensitive pagerank*, in *Proceedings of the 11th international conference on World Wide Web*, pp. 517–526, ACM, 2002.

[82] D. Chen, A. Fisch, J. Weston, and A. Bordes, *Reading wikipedia to answer open-domain questions*, 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*, pp. 1870–1879, 2017.

[83] B. Yang and T. Mitchell, *Leveraging knowledge bases in lstms for improving machine reading*, in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, pp. 1436–1446, 2017.

[84] T. Mihaylov and A. Frank, *Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge*, in *ACL 2018*, pp. 821–832, 2018.

[85] 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.

[86] A. H. Miller, A. Fisch, J. Dodge, A. 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, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pp. 1400–1409, 2016.

[87] T. N. Kipf and M. Welling, *Semi-supervised classification with graph convolutional networks*, *CoRR* **abs/1609.02907** (2016) [arXiv:1609.0290].

[88] M. Yu, W. Yin, K. S. Hasan, C. N. 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, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pp. 571–581, 2017.

[89] S. Wang, M. Yu, J. Jiang, W. Zhang, X. Guo, S. Chang, Z. Wang, T. Klinger, G. Tesauro, and M. Campbell, *Evidence aggregation for answer re-ranking in open-domain question answering*, in *6th International Conference on Learning Representations, ICLR 2018*, (Vancouver, British Columbia, Canada), 2018.

[90] W. Yang, Y. Xie, A. Lin, X. Li, L. Tan, K. Xiong, M. Li, and J. Lin, *End-to-end open-domain question answering with BERTserini*, in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, (Minneapolis, Minnesota), pp. 72–77, Association for Computational Linguistics, June, 2019.

[91] W. Xiong, J. Du, W. Y. Wang, and V. Stoyanov, *Pretrained encyclopedia: Weakly supervised knowledge-pretrained language model*, in *8th International Conference on Learning Representations, ICLR 2020*, (Addis Ababa, Ethiopia), 2020.

[92] S. Min, D. Chen, H. Hajishirzi, and L. Zettlemoyer, *A discrete hard EM approach for weakly supervised question answering*, in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, (Hong Kong, China), pp. 2851–2864, Association for Computational Linguistics, Nov., 2019.

[93] S. Wang, M. Yu, X. Guo, Z. Wang, T. Klinger, W. Zhang, S. Chang, G. Tesauro, B. Zhou, and J. Jiang, *R^3: Reinforced ranker-reader for open-domain question answering*, in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, AAAI 2018* (S. A. McIlraith and K. Q. Weinberger, eds.), (New Orleans, Louisiana, USA), pp. 5981–5988, AAAI Press, 2018.

[94] Y. Lin, H. Ji, Z. Liu, and M. Sun, *Denoising distantly supervised open-domain question answering*, in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, (Melbourne, Australia), pp. 1736–1745, Association for Computational Linguistics, July, 2018.

[95] S. E. Robertson and H. Zaragoza, *The probabilistic relevance framework: BM25 and beyond*, *Foundations and Trends in Information Retrieval* **3** (2009), no. 4 333–389.

[96] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, *A large annotated corpus for learning natural language inference*, in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, (Lisbon, Portugal), pp. 632–642, Association for Computational Linguistics, Sept., 2015.

[97] A. Parikh, O. Täckström, D. Das, and J. Uszkoreit, *A decomposable attention model for natural language inference*, in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, (Austin, Texas), pp. 2249–2255, Association for Computational Linguistics, Nov., 2016.

[98] Q. Chen, Z.-H. Ling, and X. Zhu, *Enhancing sentence embedding with generalized pooling*, in *Proceedings of the 27th International Conference on Computational Linguistics*, (Santa Fe, New Mexico, USA), pp. 1815–1826, Association for Computational Linguistics, Aug., 2018.

[99] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *BERT: Pre-training of deep bidirectional transformers for language understanding*, in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019*, (Minneapolis, Minnesota, USA), pp. 4171–4186, Association for Computational Linguistics, 2019.

[100] K. Lee, M.-W. Chang, and K. Toutanova, *Latent retrieval for weakly supervised open domain question answering*, in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, (Florence, Italy), pp. 6086–6096, Association for Computational Linguistics, July, 2019.

[101] W. Chang, F. X. Yu, Y. Chang, Y. Yang, and S. Kumar, *Pre-training tasks for embedding-based large-scale retrieval*, in *8th International Conference on Learning Representations, ICLR 2020*, (Addis Ababa, Ethiopia), 2020.

[102] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang, *REALM: retrieval-augmented language model pre-training*, *CoRR* **abs/2002.08909** (2020) [arXiv:2002.0890].

[103] E. M. Voorhees, *The TREC-8 question answering track report*, in *Proceedings of The Eighth Text REtrieval Conference, TREC 1999* (E. M. Voorhees and D. K. Harman, eds.), (Gaithersburg, Maryland, USA), National Institute of Standards and Technology, 1999.

[104] C. C. T. Kwok, O. Etzioni, and D. S. Weld, *Scaling question answering to the web*, in *Proceedings of the Tenth International World Wide Web Conference, WWW 2001* (V. Y. Shen, N. Saito, M. R. Lyu, and M. E. Zurko, eds.), (Hong Kong, China), pp. 150–161, ACM, 2001.

[105] E. Brill, S. T. Dumais, and M. Banko, *An analysis of the askmsr question-answering system*, in *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing, EMNLP 2002*, (Philadelphia, Pennsylvania, USA), 2002.

[106] D. A. Ferrucci, E. W. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. M. Prager, N. Schlaefer, and C. A. Welty, *Building watson: An overview of the deepqa project*, *AI Magazine* **31** (2010), no. 3 59–79.

[107] P. Baudiš, *Yodaqa: a modular question answering system pipeline*, in *19th International Student Conference on Electrical Engineering*, pp. 1156–1165, 2015.

[108] C. Clark and M. Gardner, *Simple and effective multi-paragraph reading comprehension*, in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018* (I. Gurevych and Y. Miyao, eds.), (Melbourne, Australia), pp. 845–855, Association for Computational Linguistics, 2018.

[109] Z. Wang, P. Ng, X. Ma, R. Nallapati, and B. Xiang, *Multi-passage BERT: A globally normalized BERT model for open-domain question answering*, in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019* (K. Inui, J. Jiang, V. Ng, and X. Wan, eds.), (Hong Kong, China), pp. 5877–5881, Association for Computational Linguistics, 2019.

[110] J. Lee, S. Yun, H. Kim, M. Ko, and J. Kang, *Ranking paragraphs for improving answer recall in open-domain question answering*, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (E. Riloff,

D. Chiang, J. Hockenmaier, and J. Tsujii, eds.), (Brussels, Belgium), pp. 565–569, Association for Computational Linguistics, 2018.

[111] Y. Zhang, D. Shen, G. Wang, Z. Gan, R. Henao, and L. Carin, *Deconvolutional paragraph representation learning*, in *Thirty-first Annual Conference on Neural Information Processing Systems, NIPS 2017* (I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, eds.), (Long Beach, California, USA), pp. 4169–4179, 2017.

[112] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, *Supervised learning of universal sentence representations from natural language inference data*, in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, (Copenhagen, Denmark), pp. 670–680, Association for Computational Linguistics, Sept., 2017.

[113] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, *Attention is all you need*, in *Thirty-first Annual Conference on Neural Information Processing Systems, NIPS 2027* (I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, eds.), (Long Beach, California, USA), pp. 5998–6008, 2017.

[114] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, *BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension*, *CoRR* **abs/1910.13461** (2019) [arXiv:1910.1346].

[115] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. P. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, K. Toutanova, L. Jones, M. Kelcey, M. Chang, A. M. Dai, J. Uszkoreit, Q. Le, and S. Petrov, *Natural questions: a benchmark for question answering research*, *Transactions of the Association for Computational Linguistics* **7** (2019) 452–466.

[116] X. Ma, Q. Zhu, Y. Zhou, X. Li, and D. Wu, *Improving question generation with sentence-level semantic matching and answer position inferring*, in *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, (New York, New York, USA), AAAI Press, 2020.

[117] L. Logeswaran and H. Lee, *An efficient framework for learning sentence representations*, in *6th International Conference on Learning Representations, ICLR 2018*, (Vancouver, British Columbia, Canada), 2018.

[118] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, *Curriculum learning*, in *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009* (A. P. Danyluk, L. Bottou, and M. L. Littman, eds.), (Montreal, Québec, Canada), pp. 41–48, ACM, 2009.

[119] M. P. Kumar, B. Packer, and D. Koller, *Self-paced learning for latent variable models*, in *24th Annual Conference on Neural Information Processing Systems, NIPS 2010* (J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, eds.), (Vancouver, British Columbia, Canada), pp. 1189–1197, Curran Associates, Inc., 2010.

[120] L. Jiang, D. Meng, Q. Zhao, S. Shan, and A. G. Hauptmann, *Self-paced curriculum learning*, in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence* (B. Bonet and S. Koenig, eds.), (Austin, Texas, USA), pp. 2694–2700, AAAI Press, 2015.

[121] A. Asai, K. Hashimoto, H. Hajishirzi, R. Socher, and C. Xiong, *Learning to retrieve reasoning paths over wikipedia graph for question answering*, in *8th International Conference on Learning Representations, ICLR 2020*, (Addis Ababa, Ethiopia), 2020.

[122] S. Min, D. Chen, L. Zettlemoyer, and H. Hajishirzi, *Knowledge guided text retrieval and reading for open domain question answering*, *CoRR* **abs/1911.03868** (2019) [arXiv:1911.0386].

[123] 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, EMNLP 2013*, (Seattle, Washington, USA), pp. 1533–1544, ACL, 2013.

[124] P. Baudis and J. Sedivý, *Modeling of the question answering task in the yodaqa system*, in *6th International Conference of the CLEF Association, CLEF 2015* (J. Mothe, J. Savoy, J. Kamps, K. Pinel-Sauvagnat, G. J. F. Jones, E. SanJuan, L. Cappellato, and N. Ferro, eds.), (Toulouse, France), pp. 222–228, Springer, 2015.

[125] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, *Squad: 100, 000+ questions for machine comprehension of text*, in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016* (J. Su, X. Carreras, and K. Duh, eds.), (Austin, Texas, USA), pp. 2383–2392, The Association for Computational Linguistics, 2016.

[126] M. Joshi, E. Choi, D. S. Weld, and L. Zettlemoyer, *Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension*, in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017* (R. Barzilay and M. Kan, eds.), (Vancouver, Canada), pp. 1601–1611, Association for Computational Linguistics, 2017.

[127] J. Johnson, M. Douze, and H. Jégou, *Billion-scale similarity search with gpus*, *IEEE Transactions on Big Data* (2019).

[128] A. Roberts, C. Raffel, and N. Shazeer, *How much knowledge can you pack into the parameters of a language model?*, *CoRR* **abs/2002.08910** (2020) [arXiv:2002.0891].

[129] V. Karpukhin, B. Oguz, S. Min, L. Wu, S. Edunov, D. Chen, and W. Yih, *Dense passage retrieval for open-domain question answering*, *CoRR* **abs/2004.04906** (2020) [arXiv:2004.0490].

[130] Y. Luan, J. Eisenstein, K. Toutanova, and M. Collins, *Sparse, dense, and attentional representations for text retrieval*, *CoRR* **abs/2005.00181** (2020) [arXiv:2005.0018].

[131] M. J. Seo, J. Lee, T. Kwiatkowski, A. P. Parikh, A. Farhadi, and H. Hajishirzi, *Real-time open-domain question answering with dense-sparse phrase index*, in *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019* (A. Korhonen, D. R. Traum, and L. Màrquez, eds.), (Florence, Italy), pp. 4430–4441, Association for Computational Linguistics, 2019.

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

[133] J. Devlin, M. Chang, K. Lee, and K. Toutanova, *BERT: pre-training of deep bidirectional transformers for language understanding*, in *NAACL-HLT (1)*, pp. 4171–4186, ACL, 2019.

[134] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, *Xlnet: Generalized autoregressive pretraining for language understanding*, *arXiv preprint arXiv:1906.08237* (2019).

[135] M. E. Peters, M. Neumann, L. Zettlemoyer, and W. Yih, *Dissecting contextual word embeddings: Architecture and representation*, in *EMNLP*, pp. 1499–1509, Association for Computational Linguistics, 2018.

[136] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning, *What does bert look at? an analysis of bert's attention*, *arXiv preprint arXiv:1906.04341* (2019).

[137] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, *GLUE: A multi-task benchmark and analysis platform for natural language understanding*, in *ICLR*, 2019.

[138] S. Zhang, X. Liu, J. Liu, J. Gao, K. Duh, and B. Van Durme, *Record: Bridging the gap between human and machine commonsense reading comprehension*, *arXiv preprint arXiv:1810.12885* (2018).

[139] R. Zellers, Y. Bisk, R. Schwartz, and Y. Choi, *SWAG: A large-scale adversarial dataset for grounded commonsense inference*, in *EMNLP*, pp. 93–104, ACL, 2018.

[140] R. Logan, N. F. Liu, M. E. Peters, M. Gardner, and S. Singh, *Barack's wife hillary: Using knowledge graphs for fact-aware language modeling*, in *ACL*, pp. 5962–5971, ACL, 2019.

[141] F. Petroni, T. Rocktäschel, P. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, and S. Riedel, *Language models as knowledge bases?*, *EMNLP* (2019).

[142] Z. Zhang, X. Han, Z. Liu, X. Jiang, M. Sun, and Q. Liu, *ERNIE: enhanced language representation with informative entities*, in *ACL (1)*, pp. 1441–1451, ACL, 2019.

[143] M. E. Peters, M. Neumann, I. Logan, L. Robert, R. Schwartz, V. Joshi, S. Singh, and N. A. Smith, *Knowledge enhanced contextual word representations*, *EMNLP* (2019).

[144] Z. Wang, P. Ng, X. Ma, R. Nallapati, and B. Xiang, *Multi-passage bert: A globally normalized bert model for open-domain question answering*, *arXiv preprint arXiv:1908.08167* (2019).

[145] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, *Distributed representations of words and phrases and their compositionality*, in *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[146] J. Pennington, R. Socher, and C. D. Manning, *Glove: Global vectors for word representation*, in *EMNLP*, pp. 1532–1543, ACL, 2014.

[147] B. McCann, J. Bradbury, C. Xiong, and R. Socher, *Learned in translation: Contextualized word vectors*, in *NIPS*, pp. 6294–6305, 2017.

[148] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, *Language models are unsupervised multitask learners*, *OpenAI Blog* **1** (2019), no. 8.

[149] C. J. Fillmore *et. al.*, *Frame semantics and the nature of language*, in *Annals of the New York Academy of Sciences: Conference on the origin and development of language and speech*, 1976.

[150] M. Minsky, *Society of mind*. Simon and Schuster, 1988.

[151] S. Ahn, H. Choi, T. Pärnamaa, and Y. Bengio, *A neural knowledge language model*, *arXiv preprint arXiv:1608.00318* (2016).

[152] B. Yang and T. M. Mitchell, *Leveraging knowledge bases in lstms for improving machine reading*, in *ACL (1)*, pp. 1436–1446, ACL, 2017.

[153] A. Liu, J. Du, and V. Stoyanov, *Knowledge-augmented language model and its application to unsupervised named-entity recognition*, in *NAACL-HLT (1)*, pp. 1142–1150, Association for Computational Linguistics, 2019.

[154] T. Mihaylov and A. Frank, *Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge*, in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Melbourne, Australia), pp. 821–832, Association for Computational Linguistics, July, 2018.

[155] W. Xiong, M. Yu, S. Chang, X. Guo, and W. Y. Wang, *Improving question answering over incomplete KBs with knowledge-aware reader*, in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, (Florence, Italy), Association for Computational Linguistics, July, 2019.

[156] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, *Attention is all you need*, in *NIPS*, pp. 5998–6008, 2017.

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

[158] Y. Zhu, R. Kiros, R. S. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, *Aligning books and movies: Towards story-like visual explanations by watching movies and reading books*, in *ICCV*, pp. 19–27, IEEE Computer Society, 2015.

[159] A. Bosselut, H. Rashkin, M. Sap, C. Malaviya, A. Çelikyilmaz, and Y. Choi, *COMET: commonsense transformers for automatic knowledge graph construction*, in *ACL (1)*, pp. 4762–4779, Association for Computational Linguistics, 2019.

[160] W. Xiong, T. Hoang, and W. Y. Wang, *DeepPath: A reinforcement learning method for knowledge graph reasoning*, in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, (Copenhagen, Denmark), pp. 564–573, Association for Computational Linguistics, Sept., 2017.

[161] J. Berant, A. Chou, R. Frostig, and P. Liang, *Semantic parsing on freebase from question-answer pairs*, in *EMNLP*, pp. 1533–1544, ACL, 2013.

[162] Y. Lin, H. Ji, Z. Liu, and M. Sun, *Denoising distantly supervised open-domain question answering*, in *ACL (1)*, pp. 1736–1745, Association for Computational Linguistics, 2018.

[163] M. Joshi, E. Choi, D. S. Weld, and L. Zettlemoyer, *Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension*, in *ACL (1)*, pp. 1601–1611, Association for Computational Linguistics, 2017.

[164] B. Dhingra, K. Mazaitis, and W. W. Cohen, *Quasar: Datasets for question answering by search and reading*, *arXiv preprint arXiv:1707.03904* (2017).

[165] M. Dunn, L. Sagun, M. Higgins, V. U. Guney, V. Cirik, and K. Cho, *Searchqa: A new q&a dataset augmented with context from a search engine*, *arXiv preprint arXiv:1704.05179* (2017).

[166] S. Wang, M. Yu, X. Guo, Z. Wang, T. Klinger, W. Zhang, S. Chang, G. Tesauro, B. Zhou, and J. Jiang, *R^3: Reinforced ranker-reader for open-domain question answering*, in *AAAI*, pp. 5981–5988, AAAI Press, 2018.

[167] S. Wang, M. Yu, J. Jiang, W. Zhang, X. Guo, S. Chang, Z. Wang, T. Klinger, G. Tesauro, and M. Campbell, *Evidence aggregation for answer re-ranking in open-domain question answering*, in *ICLR*, 2018.

[168] W. Yang, Y. Xie, A. Lin, X. Li, L. Tan, K. Xiong, M. Li, and J. Lin, *End-to-end open-domain question answering with bertserini*, *arXiv preprint arXiv:1902.01718* (2019).

[169] W. Yang, Y. Xie, L. Tan, K. Xiong, M. Li, and J. Lin, *Data augmentation for bert fine-tuning in open-domain question answering*, *arXiv preprint arXiv:1904.06652* (2019).

[170] K. Lee, M. Chang, and K. Toutanova, *Latent retrieval for weakly supervised open domain question answering*, in *ACL (1)*, pp. 6086–6096, Association for Computational Linguistics, 2019.

[171] 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).

[172] A. Talmor and J. Berant, *The web as a knowledge-base for answering complex questions*, in *NAACL-HLT*, pp. 641–651, Association for Computational Linguistics, 2018.

[173] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov, and C. D. Manning, *Hotpotqa: A dataset for diverse, explainable multi-hop question answering*, in *EMNLP*, pp. 2369–2380, Association for Computational Linguistics, 2018.

[174] J. Welbl, P. Stenetorp, and S. Riedel, *Constructing datasets for multi-hop reading comprehension across documents*, *Trans. Assoc. Comput. Linguistics* **6** (2018) 287–302.

[175] P. Qi, X. Lin, L. Mehr, Z. Wang, and C. D. Manning, *Answering complex open-domain questions through iterative query generation*, in *EMNLP/IJCNLP (1)*, pp. 2590–2602, Association for Computational Linguistics, 2019.

[176] Y. Nie, S. Wang, and M. Bansal, *Revealing the importance of semantic retrieval for machine reading at scale*, in *EMNLP/IJCNLP (1)*, pp. 2553–2566, Association for Computational Linguistics, 2019.

[177] S. Min, V. Zhong, L. Zettlemoyer, and H. Hajishirzi, *Multi-hop reading comprehension through question decomposition and rescoring*, in *ACL (1)*, pp. 6097–6109, Association for Computational Linguistics, 2019.

[178] C. Zhao, C. Xiong, C. Rosset, X. Song, P. N. Bennett, and S. Tiwary, *Transformer-xh: Multi-evidence reasoning with extra hop attention*, in *ICLR*, OpenReview.net, 2020.

[179] A. Asai, K. Hashimoto, H. Hajishirzi, R. Socher, and C. Xiong, *Learning to retrieve reasoning paths over wikipedia graph for question answering*, in *ICLR*, OpenReview.net, 2020.

[180] E. Perez, P. Lewis, W.-t. Yih, K. Cho, and D. Kiela, *Unsupervised question decomposition for question answering*, in *EMNLP*, Association for Computational Linguistics, 2020.

[181] J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal, *FEVER: a large-scale dataset for fact extraction and verification*, in *NAACL-HLT*, pp. 809–819, Association for Computational Linguistics, 2018.

[182] D. Chen, A. Fisch, J. Weston, and A. Bordes, *Reading Wikipedia to answer open-domain questions*, in *ACL (1)*, pp. 1870–1879, Association for Computational Linguistics, 2017.

[183] S. Wang, M. Yu, X. Guo, Z. Wang, T. Klinger, W. Zhang, S. Chang, G. Tesauro, B. Zhou, and J. Jiang, *R^3: Reinforced ranker-reader for open-domain question answering*, in *AAAI*, pp. 5981–5988, AAAI Press, 2018.

[184] K. Lee, M. Chang, and K. Toutanova, *Latent retrieval for weakly supervised open domain question answering*, in *ACL (1)*, pp. 6086–6096, Association for Computational Linguistics, 2019.

[185] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang, *Realm: Retrieval-augmented language model pre-training*, *arXiv preprint arXiv:2002.08909* (2020).

[186] V. Karpukhin, B. Oğuz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, *Dense passage retrieval for open-domain question answering*, in *EMNLP*, Association for Computational Linguistics, 2020.

[187] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, *et. al.*, *Natural questions: a benchmark for question answering research*, *Transactions of the Association for Computational Linguistics* **7** (2019) 453–466.

[188] 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.

[189] R. Das, S. Dhuliawala, M. Zaheer, and A. McCallum, *Multi-step retriever-reader interaction for scalable open-domain question answering*, *arXiv preprint arXiv:1905.05733* (2019).

[190] Y. Feldman and R. El-Yaniv, *Multi-hop paragraph retrieval for open-domain question answering*, in *ACL (1)*, pp. 2296–2309, Association for Computational Linguistics, 2019.

[191] J. Rocchio, *Relevance feedback in information retrieval*, *The Smart retrieval system-experiments in automatic document processing* (1971) 313–323.

[192] E. M. Voorhees, *Query expansion using lexical-semantic relations*, in *SIGIR'94*, pp. 61–69, Springer, 1994.

[193] I. Ruthven and M. Lalmas, *A survey on the use of relevance feedback for information access systems*, *Knowledge engineering review* **18** (2003), no. 2 95–145.

[194] W. B. Croft and D. J. Harper, *Using probabilistic models of document retrieval without relevance information*, *Journal of documentation* (1979).

[195] G. Cao, J.-Y. Nie, J. Gao, and S. Robertson, *Selecting good expansion terms for pseudo-relevance feedback*, in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 243–250, 2008.

[196] Y. Lv and C. Zhai, *Positional relevance model for pseudo-relevance feedback*, in *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pp. 579–586, 2010.

[197] K. Collins-Thompson and J. Callan, *Estimation and use of uncertainty in pseudo-relevance feedback*, in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 303–310, 2007.

[198] Y. Zhang, H. Dai, Z. Kozareva, A. J. Smola, and L. Song, *Variational reasoning for question answering with knowledge graph*, in *AAAI*, pp. 6069–6076, AAAI Press, 2018.

[199] H. Sun, T. Bedrax-Weiss, and W. W. Cohen, *Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text*, *arXiv preprint arXiv:1904.09537* (2019).

[200] B. Dhingra, M. Zaheer, V. Balachandran, G. Neubig, R. Salakhutdinov, and W. W. Cohen, *Differentiable reasoning over a virtual knowledge base*, in *ICLR*, OpenReview.net, 2020.

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

[202] V. Zhong, C. Xiong, N. S. Keskar, and R. Socher, *Coarse-grain fine-grain coattention network for multi-evidence question answering*, *arXiv preprint arXiv:1901.00603* (2019).

[203] M. Tu, G. Wang, J. Huang, Y. Tang, X. He, and B. Zhou, *Multi-hop reading comprehension across multiple documents by reasoning over heterogeneous graphs*, *arXiv preprint arXiv:1905.07374* (2019).

[204] I. Beltagy, M. E. Peters, and A. Cohan, *Longformer: The long-document transformer*, *arXiv preprint arXiv:2004.05150* (2020).

[205] 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).

[206] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, *Unsupervised feature learning via non-parametric instance discrimination*, in *CVPR*, pp. 3733–3742, IEEE Computer Society, 2018.

[207] 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* (2019) arXiv–1910.

[208] J. Johnson, M. Douze, and H. Jégou, *Billion-scale similarity search with gpus*, *arXiv preprint arXiv:1702.08734* (2017).

[209] A. Hanselowski, H. Zhang, Z. Li, D. Sorokin, B. Schiller, C. Schulz, and I. Gurevych, *Ukp-athene: Multi-sentence textual entailment for claim verification*, *arXiv preprint arXiv:1809.01479* (2018).

[210] K. Clark, M. Luong, Q. V. Le, and C. D. Manning, *ELECTRA: pre-training text encoders as discriminators rather than generators*, in *ICLR*, OpenReview.net, 2020.

[211] L. Wu, F. Petroni, M. Josifoski, S. Riedel, and L. Zettlemoyer, *Zero-shot entity linking with dense entity retrieval*, *arXiv preprint arXiv:1911.03814* (2019).

[212] T. Wolfson, M. Geva, A. Gupta, Y. Goldberg, M. Gardner, D. Deutch, and J. Berant, *Break it down: A question understanding benchmark*, *Trans. Assoc. Comput. Linguistics* **8** (2020) 183–198.

[213] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, *et. al.*, *Retrieval-augmented generation for knowledge-intensive nlp tasks*, *arXiv preprint arXiv:2005.11401* (2020).

[214] G. Izacard and E. Grave, *Leveraging passage retrieval with generative models for open domain question answering*, *arXiv preprint arXiv:2007.01282* (2020).

[215] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, *BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension*, in *ACL*, pp. 7871–7880, Association for Computational Linguistics, 2020.

[216] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, *Exploring the limits of transfer learning with a unified text-to-text transformer*, *arXiv preprint arXiv:1910.10683* (2019).

[217] Y. Fang, S. Sun, Z. Gan, R. Pillai, S. Wang, and J. Liu, *Hierarchical graph network for multi-hop question answering*, *arXiv preprint arXiv:1911.03631* (2019).

[218] P. Rajpurkar, R. Jia, and P. Liang, *Know what you don't know: Unanswerable questions for squad*, in *ACL (2)*, pp. 784–789, Association for Computational Linguistics, 2018.

[219] A. Roberts, C. Raffel, and N. Shazeer, *How much knowledge can you pack into the parameters of a language model?*, *arXiv preprint arXiv:2002.08910* (2020).

[220] Y. A. Malkov and D. A. Yashunin, *Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs*, *IEEE transactions on pattern analysis and machine intelligence* (2018).

[221] Z. Liu, C. Xiong, M. Sun, and Z. Liu, *Fine-grained fact verification with kernel graph attention network*, in *ACL*, pp. 7342–7351, Association for Computational Linguistics, 2020.

[222] J. Zhou, X. Han, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, *GEAR: graph-based evidence aggregating and reasoning for fact verification*, in *ACL (1)*, pp. 892–901, Association for Computational Linguistics, 2019.

[223] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, *Graph attention networks*, *arXiv preprint arXiv:1710.10903* (2017).