A GENERAL FOFE-NET FRAMEWORK FOR SIMPLE AND EFFECTIVE QUESTION ANSWERING OVER KNOWLEDGE BASES

DEKUN WU

A THESIS SUBMITTED TO
THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

GRADUATE PROGRAM IN COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO
APRIL 2019

Abstract

Question answering over knowledge base (KB-QA) has recently become a popular research topic in NLP. One of the popular ways to solve the KBQA problem is to make use of a pipeline of several NLP modules, including entity discovery and linking (EDL) and relation detection. Recent success on KBQA task usually involves complex network structures with sophisticated heuristics. Inspired by a previous work that builds a strong KBQA baseline, we propose a simple but general neural model composed of fixed-size ordinally forgetting encoding (FOFE) and deep neural networks, called FOFE-net to solve KB-QA problem at different stages. For evaluation, we use two popular KB-QA datasets, SimpleQuestions, WebQSP, and our newly created dataset, FreebaseQA. The experimental results show that FOFE-net performs well on KBQA subtasks, entity discovery and linking (EDL) and relation detection, and in turn pushing overall KB-QA system to achieve strong results on all the datasets.

Acknowledgements

I would like to express my special thanks of gratitude to my supervisor, Prof. Hui Jiang, for his continuous support throughout my study at York University. I am also sincerely grateful to my committee members, Prof. Manos Papagelis and Prof. Yuejiao Fu, for reviewing my thesis and providing me with their valuable advice.

I would also like to thank my senior lab mates Mingbin Xu and Hengyue Pan for sharing insight and expertise with me, which greatly assisted my research.

I would also like to acknowledge my friends Nargiza Nosirova, Bao Xin Chen and Sedtawut Watcharawittayakul for the help in the early stage of my research.

Financial support from York University is heartily acknowledged.

Finally yet importantly, I would like to thanks my beloved parents and grandmother. Their encouragement and understanding is the foremost reason and impetus for me to move forward.

Table of Contents

List of Tables

List of Figures

Abbreviations

AI	Artificial Intelligence
BOW	Bag-of-Word
CBOW	Continuous Bag-of-Words
CNN	Convolutional Neural Network
DNN	Deep Neural Network
EL	Entity Linking
EDL	Entity Discovery and Linking
FFNN	FeedForward Neural Network
FOFE	Fixed-size Ordinally Forgetting Encoding
FQ	FreebaseQA
	GRU Gated Recurrent Unit

IR	Information Retrieval
KB	Knowledge Base
KBQA	Knowledge Base Question Answering
LSTM	Long Short-Term Memory
NER	Named Entity Recognition
NL	Natural Language
NLP	Natural Language Processing
NN	Neural Network
QA	Question Answering
RD	Relation Detection
RDF	Resource Description Framework
RNN	Recurrent Neural Network
WQ	WebQSP
SGD	Stochastic Gradient Descent
SQ	SimpleQuestions

1 Introduction

1.1 Motivation

Question Answering System is capable of returning proper responses given the questions asked by the human in natural language. The earliest QA systems can be traced back to early 1960s, which tried to answer the questions about baseball games and scientific facts. These progenitors of QA systems started to use two major paradigms, information-retrieval-based and knowledge-base-based, to pull answers from an unstructured collection of textual information and structured knowledge base in response to user requests in natural language [21].

The modern QA systems pay more attention to factoid question answering. Factoid question, by definition, is the question asked about a concise fact that relate to a person, an organization, or a numerical expression etc. Questions below are examples of factoid question of which answer type is a country, a song and an American football team respectively.

- Which country hosted the 1936 Summer Olympic Games?
- Which song from Mary Poppins won the Academy Award for Best Song?
- The Chicago Bears defeated which team 73-0 in the 1940 NFL Championship game?

To answer these factoid questions, IR-based QA system uses information retrieval methods to query voluminous texts on the Web or in the collection to find the relevant documents and passages to the question, then adopt reading comprehension approaches to extract the answer from retrieved documents and passages. KB-based QA system, instead, tries to form a query over knowledge base in response to natural language request. In this thesis, we focus on KB-based question answering and its problem. Our proposed solution will be discussed in Chapter 4.

Knowledge Base Question Answering is the task that aims to answer questions over knowledge bases. The current approaches of this task can be categorized into three main streams: Semantic Parsing, Vector modeling and Information Extraction. Semantic Parsing-based methods translate natural language question into a standard logical form (e.g. lambda calculus) and then perform a query over knowledge base. Vector Modeling based methods represent questions and facts stored in knowledge bases as low dimensional vectors. Thus, the questions can be answered by computing similarity scores between these vectors.

Information Extraction based methods detect the entities conveyed in question and use these entities along with their neighbors to construct a KB sub-graph. Sub-graph nodes satisfied with specific rules can be viewed as candidate answers. Through feature extraction of candidate answers and building a classifier to measure the relevance of the question to each candidate, the final answer can be selected.

Recently, neural approaches have become popular in the KBQA field, while many of them show unnecessary complexity in dealing with KBQA task. Besides, these neural approaches are so data hungry that they arouse huge demand for more training data. In this research, we propose a simple and straightforward model to solve the KBQA task and create a new trivia-type question answering dataset for the evaluation of our model.

1.2 Contribution and Outline of the Thesis

In this thesis, we propose a Fixed-size Ordinally-Forgetting Encoding (FOFE) based approach to solve the problem and create a new dataset, FreebaseQA, to evaluate it. Our contributions are summarized as follows:

- Many previous works adopt various complex models to tackle with KBQA subtasks. Albeit promising results their models achieve, it is still arguable whether we need such complex models as most of them more or less show unnecessary complexity. In contrast, we combine FOFE, a simple representation algorithm that has been proved to be effective for many Natural Language Processing (NLP) problems, with a deep feed-forward neural network to deal with KBQA subtasks such as entity linking and relation detection. The experimental result shows FOFE-based method works well on these sub-tasks of KBQA despite its simplicity.

- We construct a stage-based KBQA system by combining FOFE-based models, and examine it on existing KBQA datasets, WebQSP and SimpleQuestions. The experimental result shows our KBQA system's performance is very competitive.

- We also create a new data set, FreebaseQA, for evaluation of our KBQA model. In comparison with other existing KBQA datasets, FreebaseQA is either larger in scale or more complex linguistically, so it would be very suitable for model training and testing in factoid QA tasks.

The rest of the thesis is organized as the following: Chapter 2 reviews the modern approaches for KBQA task. Chapter 3 reviews the text representation approaches for extracting features to feed the deep feed-forward neural network. Chapter 4 reviews the deep feed-forward neural network and how to apply it into NLP tasks. Chapter 5 gives our proposed method to combine FOFE and other encoding methods with deep feed-forward neural network in tackling with overall

KBQA task as well as its subtasks. Chapter 6 presents our new factoid QA dataset and approaches of its creation. Chapter 7 shows the experimental results of our proposed KBQA system. Chapter 8 concludes the thesis and outlines future work.

2 Modern KBQA Systems

Knowledge bases (KBs) have been intensively studied as a tool to create effective methods to answer factoid questions in natural language processing (NLP). There are several very large-scale knowledge bases [5, 11, 22] available for KB-QA tasks, which are usually organized as a graph and formatted as sets of Resource Description Framework (RDF) triples.

A RDF triple has three fields: a subject, a predicate and an object. Fig. 2.1 shows an excerpt of Freebase, an RDF knowledge base that has more than 3 billion triples, and Table 2.1 shows the corresponding RDF triples.

In many cases, people want to operate these knowledge bases in the similar way as they operate relational databases so that SPARQL [28], a semantic query language, has been come up with to enable users to access the correct resources stored in these knowledge bases. However, this query language is not very user-friendly due to its high demand on user's familiarity with its grammar as well as the structure and vocabulary of knowledge bases. Accordingly, many works

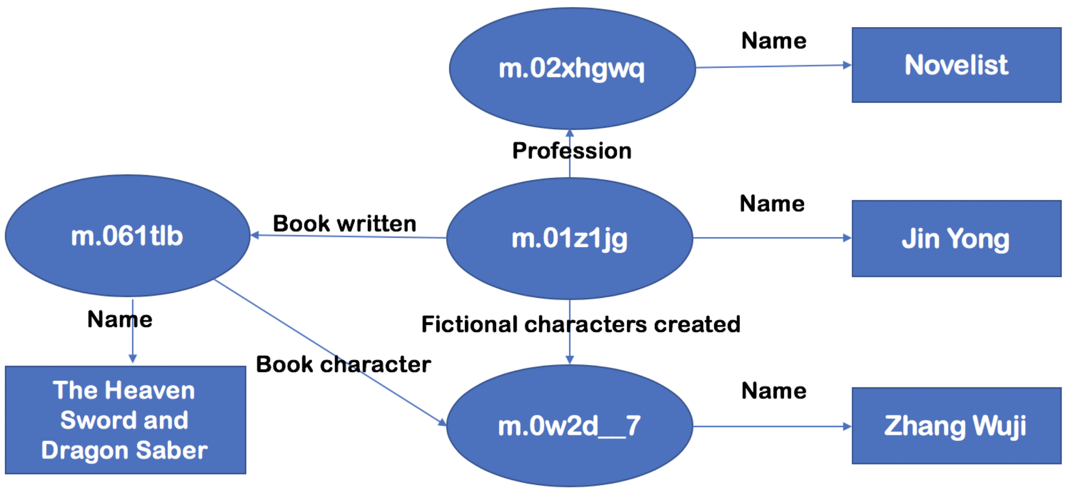


Figure 2.1: An excerpt of Freebase. The prefixes of predicates are omitted for readability.

have been done with an attempt to bridge the gap between natural languages and SPARSQL [38, 2].

With the growing interest in complex NL question answering, researchers find that it is difficult to translate the complex NL question to SPARSQL query. Instead, they have shifted to map NL question to answer via predicate calculus. Many related works have emerged in recent years [3, 4, 31, 40].

The recent success of deep learning on other Artificial Intelligence (AI) applications also inspire NLP community to apply this new technique into QA task. Bordes et al. [8] firstly introduce neural-network based methods for the KB-QA problem, which maps questions and KB triples to low-dimensional space and the

Subject	**Predicate**	**Object**
m.01z1jg	Name	Jin Yong
m.0w2d__7	Name	Zhang Wuji
m.02xhgwq	Name	Novelist
m.061tlb	Name	The Heaven Sword and Dragon Saber
m.01z1jg	Profession	m.02xhgwq
m.01z1jg	Book written	m.061tlb
m.061tlb	Book character	m.0w2d__7
m.01z1jg	Fictional character created	m.0w2d__7

Table 2.1: The set of RDF triples corresponding to the graph in Fig. 2.1.

representations of questions. The corresponding answers are made to be more similar throughout the learning process. At the test stage, the best candidate answer will be selected by computing a similarity measure between two projected vectors. Bordes et al. [6] further improve their method by enriching the representation of the answer. The new representation of answers involves question-answer path and all entities associated with the answer.

Similarly, there is an important line of research work in an end-to-end learning for the KB-QA problem. Bordes et al. [7] introduce a new single-relation KB-QA benchmark dataset (SimpleQuestions) and apply memory network into this KB-

QA task. They represent the facts of KB as a bag-of-symbol vector and questions as a bag-of-ngrams vector. Then the answer would be returned based on the best cosine similarity of a supporting fact in KB with the question. He and Golub [17] propose a character-level encoder-decoder framework enhanced by the attention mechanism for single-relation question answering. This framework uses character-level Long Short-Term Memory (LSTM) and character-level Convolutional Neural Networks (CNNs) to encode the questions and the pairs of predicates and entities in the KB respectively. The decoder in this framework is an LSTM with attention mechanism used to generate the topic entity and predicate. Lukovnikov et al. [24] use a Gated Recurrent Unit (GRU) based model to encode questions and entities on both character and word level and to encode the relations on the word level. The answers might be found by computing the cosine similarities between the question and the entity, and between the question and the predicate [16].

Moreover, some previous works divide the KB-QA task into several NLP subtasks: entity discovery, entity linking, relation detection, and constraint detection (optional) [10, 42, 36, 43, 30, 15]. Figure 2.2 shows the pipeline of such KB-QA systems. Firstly, named entity detector is responsible for locating topic subject mentions in the question. Secondly, an entity linker will link the detected topic subject mention to the entity node in the knowledge base. Next, relation detector is used to choose the right paths from the entity node to search for the answers. At

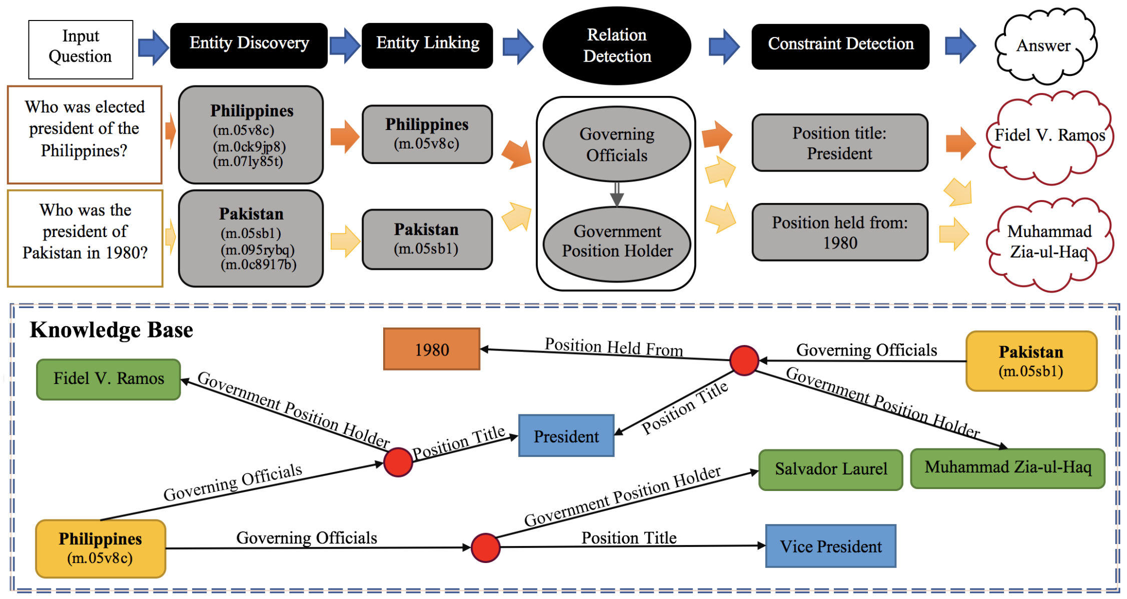


Figure 2.2: The general pipeline to solve KB-QA problems, adopt in this thesis. Given input questions: *Who was elected president of the Philippines?*, the named entity detector should detect **topic subject mention**: {*Philippines*} in question, and entity linker further links this mention to **subject entity**: {*m.05v8c*}. Relation detector detects **relation / inferential chain** implied in the question, which is {*Governing Officials, Government Position Holder*} in this case. A set of candidate answers can be queried by using the predicted subject-relation pair. With the **constraint**: {*President*} and {*Past*} (not showing in this Figure) get detected, only former presidents of the Philippines will be returned as the final answers.

last, the constraint detector imposes the potential constraints based on the question, which will be used to further prune the found paths and produce a set of answers that may best match the question.

Semantic parsing approaches have also been developed in the era of deep learning. For example, Yih et al. [40] propose a new semantic parser with deep Convolutional Neural Networks (CNN) for identifying the core inferential chain in the NL question.

Despite the exciting improvement in the KBQA task we have seen recently, the neural network structures go increasingly complex. Mohammed et al. [26] build a strong baseline for KB-QA task and prove through experiments that best models at most provide modest gains over their baselines and some of those models show unnecessary complexity. Following their findings, we are curious to look for a simple but stronger architecture that is enabled to compete with the state-of-the-art models on KB-QA task.

Recently, fixed-size ordinally forgetting encoding (FOFE) method [44] has been proposed to model variable-length sequences into fixed-size representations in a lossless way. This simple ordinally-forgetting mechanism has been applied to some NLP tasks, e.g. language modeling [44, 35], word embeddings [32], named entity recognition (NER) [37], and has achieved very competitive results. Due to its practical simplicity and technical soundness, we extend this method to KB-QA task, which will be introduced in Chapter 5.

3 Text Representation

In this chapter, we introduce how to encode a variable-length text to a fixed-size dense vector, which can be used as features to train the Deep Neural Network (DNN).

3.1 Bag-of-Words (BOW)

Bag-of-Words (BOW) is a very simple and straightforward way to model a variable-length text (such as a sentence or a document). This representation only keeps the word frequency in the text, with disregard for word order and syntax. Suppose we have two texts "*He joined NBA in 2003 and won his first NBA championship in 2012.*", "*He won his second NBA championship in 2013.*" and a vocabulary [*in*, *2012*, *2013*, *He*, *his*, *first*, *second*, *NBA*, *championship*, *won*, *2003*, *joined*, *and*], then we can represent those text as bag-of-words (See Table 3.1). We can only keep the second column and the third column in the table and view them as vector representations, then we have [*2,1,0,1,1,1,0,2,1,1,1,1,1*] to represent the

Word	**Frequency in T1**	**Frequency in T2**
in	2	1
2012	1	0
2013	0	1
He	1	1
his	1	1
first	1	0
second	0	1
NBA	2	1
championship	1	1
won	1	1
2003	1	0
joined	1	0
and	1	0

Table 3.1: Bag-of-Words representations of two texts " He joined NBA in 2003 and won his first NBA championship in 2012." and "He won his second NBA championship in 2013.".

text "*He joined NBA in 2003 and won his first NBA championship in 2012.*" and [*1,0,1,1,1,0,1,1,1,1,0,0,0*] to represent the text "*He won his second NBA championship in 2013.*". Note that albeit different length of original texts, their text representations have the same dimension $|V|$, the vocabulary size. However, these vector representations are too sparse to train the neural network, In Section 4.5, we will show how to convert these sparse vectors to condense ones by a simple matrices multiplication. Then they will be suitable for training the neural network.

3.2 TF-IDF

TF-IDF is another algorithm to model a collection of text (such as a sentence or a document). This algorithm is a product of two terms, **term frequency** and **inverse document frequency**.

Term frequency t_{ij} is the frequency of term i in document j. Usually we want to downweight the importance of term that has a high frequency, so logarithmic term frequency is used to substitute for the raw one, which is presented below:

$$t_{ij} = \begin{cases} 1 + \boldsymbol{log_{10} count(i,j)}, & \text{if } count(i,j) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

Inverse document frequency is used to assign importance to the words that can only be found in a few documents. Because these words are more informative and helpful for discriminating these documents from the rest. The definition of inverse document frequency is:

$$idf_i = log_{10}\frac{N+1}{df_i+1} + 1 \tag{3.2}$$

where N is the total number of documents in the collection, df_i is the number of documents that contains term i.

Then the tf-idf weight w_ij for the term i in a document j can be calculated as the product of term frequency and inverse document frequency:

$$w_ij = t_{ij} \times idf_i \tag{3.3}$$

we use L2 norm to normalize the w_ij into the range $[0, 1]$, then we have:

$$w_{norm_{ij}} = \frac{w_{ij}}{\sqrt{\sum_i w_{ij}^2}} \tag{3.4}$$

Let's use an example to show what the tf-idf weighted vector of documents would look like. Suppose we have three documents in the collection:

Document D1: "I got the answer"

Document D2: "What is the answer to this question"

Document D3: "I do not know the answer"

Their tf-idf weighted vectors are shown in Table 3.2. Similar to Bag-of-Words algorithm, TF-IDF method can also represent a variable-length text to a vector of length $|V|$.

Word	**D1**	**D2**	**D3**
answer	0.391	0.247	0.286
do	0.0	0.0	0.484
got	0.663	0.0	0.0
I	0.504	0.0	0.368
is	0.0	0.419	0.0
know	0.0	0.0	0.484
not	0.0	0.0	0.484
question	0.0	0.419	0.0
the	0.391	0.247	0.286
this	0.0	0.419	0.0
to	0.0	0.419	0.0
What	0.0	0.419	0.0

Table 3.2: The TF-IDF representations for Document D1, Document D2 and Document D3.

3.3 Fixed-size Ordinally Forgetting Encoding

Fixed-size ordinally forgetting encoding (FOFE) is proposed to encode any sequence of variable length into a fixed-size representation [44]. It has shown that the FOFE encoding is lossless and unique.

Let V be a vocabulary set, with each word represented as a one-hot vector. FOFE extends bag-of-words (BoW) by incorporating a forgetting factor to capture positional information. Let $S = w_1, w_2, \cdots, w_T$ represent a sequence of T words, and e_t represent the one-hot vector of the t-th word in S, where $1 \leq t \leq T$. Then the FOFE encoding of the partial sequence $w_1 w_2 \cdots w_t$ is defined recursively as follows:

$$\boldsymbol{z_t} = \begin{cases} \mathbf{0}, & \text{if } t = 0 \\ \alpha \cdot \boldsymbol{z_{t-1}} + \boldsymbol{e_t}, & \text{otherwise} \end{cases} \tag{3.5}$$

where α denotes the forgetting factor picked between 0 and 1. Same with bag-of-words and tf-idf, the size of $\boldsymbol{z_t}$ is fixed at $|V|$.

For example, assume a vocabulary V containing the words *"I"*, *"think"*, *","*, *"therefore"*, *"am"*, with corresponding one-hot vectors $[1, 0, 0, 0, 0]$, $[0, 1, 0, 0, 0]$, $[0, 0, 1, 0, 0]$, $[0, 0, 0, 1, 0]$ and $[0, 0, 0, 0, 1]$ respectively. Then the FOFE encoding of the partial sequences of *"I think, therefore I am"* are shown in the Table 3.3.

It is shown that the encoded word sequences can be unequivocally recovered [44].

Partial Sequence	**FOFE**
I	1,0,0,0,0
I think	α, 1, 0, 0, 0
I think,	α^2, α, 1, 0, 0
I think, therefore	α^3, α^2, α, 1, 0
I think, therefore I	$\alpha^4 + 1$, α^3, α^2, α, 0
I think, therefore I am	$\alpha^5 + \alpha$, α^4, α^3, α^2, 1

Table 3.3: Partial encoding of *"I think, therefore I am"*

The uniqueness of the FOFE encoding is guaranteed by the following two theorems:

Theorem 1. *For $0 < \alpha \leq 0.5$, FOFE is unique for any countable vocabulary V and any finite value T.*

Theorem 2. *For $0.5 < \alpha < 1$, given any finite value T and any countable vocabulary V, FOFE is unique almost everywhere, except only a finite set of countable choices of α.*

The uniqueness is not guaranteed when α is chosen to be between 0.5 and 1, the chance of coming across such scenarios is very slim due to quantization errors in the system. Therefore, we can confidently assume that FOFE can uniquely encode any sequence of words of arbitrary length, yielding fixed-size and lossless representations.

3.4 Word2Vec

In the sections above, we introduce three algorithms to encode a variable-length text to a fixed-length vector. However, these algorithms result in sparse representations, which means most of their elements are zero. In NLP field, we prefer dense representation than sparse representation since it requires fewer parameters in training and helps to avoid overfitting.

Similar to the way that texts can be represented as vectors, semantics of words can also be encoded into a vector space to capture the closeness of each word in the vocabulary. Word2Vec is an algorithm to produce a dense vector of a word [25]. By combining this method with one of the above three text representation methods, a variable-length text can be represented as a dense vector.

The intuition of word2vec is very simple, it trains a classifier to predict the likelihood of a word w_i occurs near the word w_j. By doing the prediction task, the embedding representation for each word is learned as well.

Consider a set S^+ of correct word-context pairs and a set S^- of the incorrect word-context pairs. The goal of this algorithm is to maximize the following objective function:

$$L(\theta) = \sum_{(w,c)\in S^+} \log P(1|w,c) + \sum_{(w,c)\in S^-} \log P(0|w,c) \tag{3.6}$$

where (w, c) is a word-context pair.

Normally, the probability function $P(1|w, c)$ is a sigmoid function over similarity score $s(w, c)$:

$$P(1|w,c) = \frac{1}{1 + e^{-s(w,c)}} \tag{3.7}$$

$s(w, c)$ is known as cosine similarity, which is widely used to measure the similarity between word w and context word c, which can simply be a normalized dot product of two vectors:

$$s(w,c) = \frac{w.c}{\|w\| . \|c\|} \tag{3.8}$$

We can see that the cosine similarity is based on the dot product, which tends to be large when the two vectors have large values in the same dimensions. But the raw dot product is problematic since it favors frequent words that have long vectors. To solve the problem, people tend to normalize the raw dot product of the two vectors by dividing it by the lengths of each of the two vectors. (See Equation 3.8).

Interestingly, another common similarity metric, Squared Euclidean Distance, is proportional to the cosine similarity (See Equation 3.9). In practice, these two types of metrics can be used interchangeably, as maximizing the cosine similarity of the two vectors is equivalent to minimize the squared euclidean distance between them.

$$E(w,c) = \sqrt{\left(\frac{w}{\|w\|} - \frac{c}{\|c\|}\right)^2} = \sqrt{\left(\frac{w}{\|w\|}\right)^2 + \left(\frac{c}{\|c\|}\right)^2 - \frac{2 \times w.c}{\|w\| . \|c\|}} = \sqrt{2 - 2 \times s(w,c)} \qquad (3.9)$$

Given two random initialized $V \times d$ matrices W and C. d is a parameter that is usually set to be in the range from 100 to 500. Each row of these matrices is a dense representation of a word, and they will be updated during the training process. Finally, words having similar contexts are expected to have similar vectors, such as "cat" and "dog". However, "dog" and "flag" should be far from each other since they are semantically unrelated.

Once matrices W and C are converged, we can add these two embeddings together as a new d-dimensional word embedding U.

3.5 Dense Text Representation

As discussed above, Bag-of-Words, TF-IDF and FOFE can represent an arbitrary length text to a sparse vector of length $|V|$, which we denote as E. Word2Vec can produce a $V \times d$ word embedding matrix U, each row of which is a dense representation of a word. We want to project the high dimensional sparse vector of texts onto a low dimensional space, which can be done by a simple matrices multiplication:

$$M = EU \tag{3.10}$$

where M is a $1 \times d$ dense vector so that it can be used as a feature to train the deep feedforward neural network. In Chapter 5, we will show how to combine the feedforward neural network with this dense text representation to solve the KBQA task as well as its subtasks.

4 Deep Neural Network

In this chapter, we review the concept of Deep Feedfoward Neural Network and how to apply this architecture into NLP tasks.

4.1 Deep Feedforward Neural Network

Deep Feedforward Neural Network is a common kind of neural network, which is acyclic, fully-connected, multilayer and always passes the computing results from the units in preceding layers to the units in succeeding layers. The goal of a Feedforward Neural Network is to approximate a function f^* that maps input x to target t [14].

Recall that each layer of FFNN consists of a number of hidden units, each of which has a weight vector w and a bias scalar b. We denote the weight of the connection from the i-th input unit xi to the the j-th hidden unit hj in the n-th layer as W^n_{ij}. The weight matrix for the n-th layer can be represented as $W^{(n)}$ by combining weight W_{ij} for every input unit i and hidden unit j pair. Similarly, bias

b_i for unit i in the n-th layer can be combined as vector b^n. Let a^i be the output from layer i, and z^i be the combination of weights and biases $W^i a^{i-1} + b^i$. The algorithm for computing the result of an n-layer feedforward neural network in the forward step is presented below:

```
   ▷ $a^0 = x$
1: for $i \leftarrow 1...n$ do
2:     $z^i = W^i a^{i-1} + b^i$
3:     $a^i = \phi^i(z^i)$
4: end for
5: $\widehat{y} = a^n$
```

$\widehat{y}$ is the estimation made by a feedforward neural network. Assume there is a function f^* that can perfectly fit every given input-target pair (x, t). We want feedforward neural network can learn a function f that estimate $\widehat{y}$ as similar as possible to gold target t given certain input x. That is, in other words, the function f is expected to be as similar as possible to the function f^*. To realize this goal, we must update two learnable parameters: the weight matrix $W^{(n)}$ and the bias vector b^n, iteratively during the training process. The learning process will be discussed in the following section.

4.2 Training DNN

In Sec. 4, we introduce how the information flow through the feedforward neural network from the input x to produce the output $\widehat{y}$. Once the output $\widehat{y}$ is produced, a loss function will be used to measure how incorrect the output $\widehat{y}$ is given the target t. Assume this is a binary classification task, which means the target t can either be 1 or 0. Then we can choose loss function to be Cross Entropy Loss:

$$L_{CE}\left(\widehat{y}, t\right) = -\left[t \log \widehat{y} + \left(1 - t\right) \log\left(1 - \widehat{y}\right)\right] \tag{4.1}$$

or Hinge Loss:

$$L_{HL} = \max\left(0, \gamma + \widehat{y}_{-} - \widehat{y}_{+}\right) \tag{4.2}$$

where $\widehat{y}_{-}$ is the estimation value $\forall t = 0$, $\widehat{y}_{+}$ is the estimation value for $t = 1$, γ is a constant number.

Once the loss is produced, the information will flow backward through the network to compute the gradient of the loss function with respect to the parameters. This step is called Back-propagation.

Let $\nabla_{\theta} L\left(f\left(x; \theta\right), t\right)$ be the gradient of the loss function with respect to the parameters θ. Optimization algorithm such as Stochastic Gradient Descent (SGD) can be used to minimize the loss function by computing $\nabla_{\theta} L\left(f\left(x; \theta\right), t\right)$ after each training iteration. Suppose the training data consist of n tuples $(x^1, t^1), (x^2, t^2), ...,$ (x^n, t^n), we can use the following SGD algorithm to learn the neural network:

```
1: function STOCHASTIC GRADIENT DESCENT($x$,$t$,$\eta$)
   ▷ $\eta$: Learning Rate
   ▷ Randomly initialize parameters $\theta$
2:     while $\theta$ not converged do
3:         Randomly shuffle examples in the training set.
4:         for $i \leftarrow 1...n$ do
5:             $\theta \leftarrow \theta - \eta \nabla_\theta L\left(f\left(x^i;\theta\right), t^i\right)$
6:         end for
7:     end while
8:     return $\theta$
9: end function
```

4.3 Apply DNN Architecture Into NLP Tasks

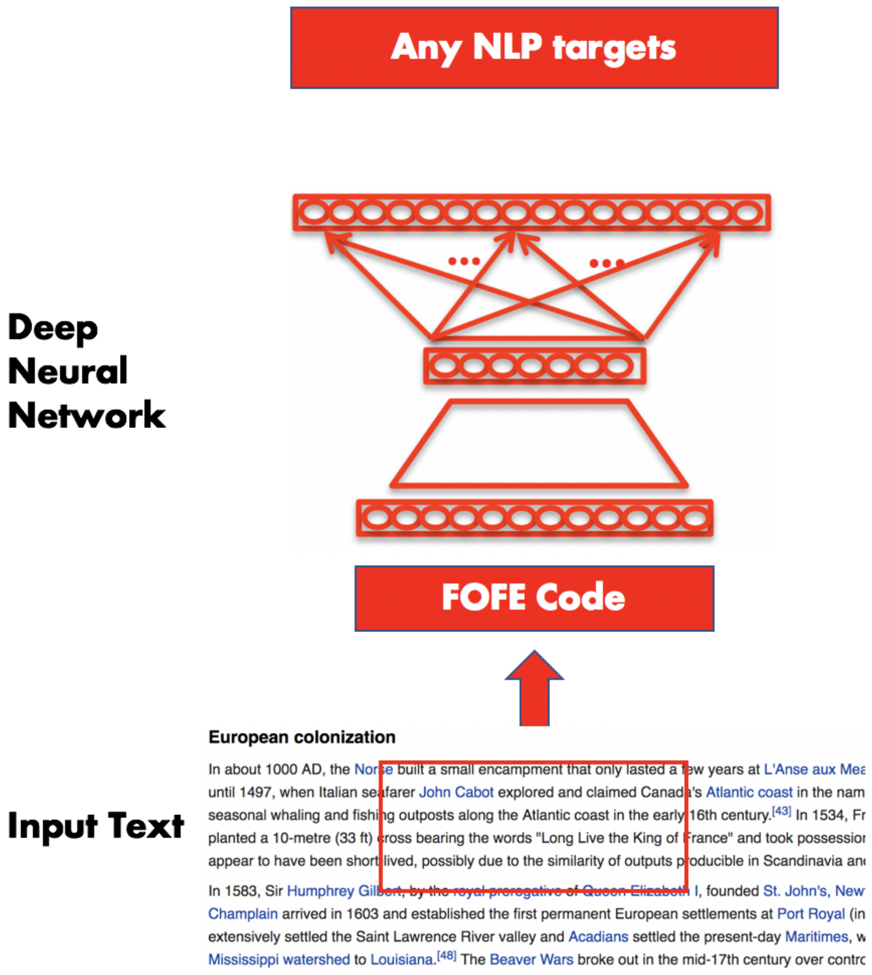
As discussed above, Feedforward deep neural networks (DNNs) use rather simple structure consisting of several fully-connected layers to perform as a powerful universal approximators for mapping any given inputs to desired outputs [18]. It is very natural to apply this architecture into NLP tasks, which is expected to make predictions based on textual information.

However, a major deficiency of feedforward deep neural network in tackling with NLP tasks is that this architecture requires fixed size input while we are interested in making prediction based on texts of varying length (a sequence of letters in the sentence, a sequence of words in the sentence, a sequence of sentence in the paragraph and so on) in most cases.

A traditional way to solve this problem is to feed Continuous Bag-of-Words (CBOW) representation into the feedforward deep neural network. But this method suffers from losing useful positional information [13].

Recurrent Neural Network and Convolutional Neural Network are two variants of Neural Network that are capable of extracting fixed size representation from the text of arbitrary length without losing useful ordering information. Hence they can be integrated with feedforward neural network for the prediction tasks. But the drawback of building such hierarchical structure is that this would involve more parameters to train and the training time would increase as the error has to go backward from the DNN for the final task through the RNN or CNN for the text representation.

A better solution is to combine DNNs with the fixed-size ordinally forgetting encoding (FOFE) method. FOFE is used as a frontend encoding method to convert any sequence of words in an NLP task into a fixed-size representation without losing



European colonization

In about 1000 AD, the Norse built a small encampment that only lasted a few years at L'Anse aux Mea
until 1497, when Italian seafarer John Cabot explored and claimed Canada's Atlantic coast in the nam
seasonal whaling and fishing outposts along the Atlantic coast in the early 16th century.[43] In 1534, Fr
planted a 10-metre (33 ft) cross bearing the words "Long Live the King of France" and took possessior
appear to have been short-lived, possibly due to the similarity of outputs producible in Scandinavia an

In 1583, Sir Humphrey Gilbert, by the royal prerogative of Queen Elizabeth I, founded St. John's, New
Champlain arrived in 1603 and established the first permanent European settlements at Port Royal (in
extensively settled the Saint Lawrence River valley and Acadians settled the present-day Maritimes, w
Mississippi watershed to Louisiana.[48] The Beaver Wars broke out in the mid-17th century over contr

Figure 4.1: The FOFE-net framework.

any information and the simple DNNs are used as universal function approximators to map these fixed-size representations into any desirable NLP target labels. This framework, called FOFE-net (See Figure 4.1), may be appealing to many NLP tasks since FOFE is simple and fast and requires neither learning nor feature engineering (except a single hyperparameter), and DNNs are also much easier and faster to manipulate than RNNs and CNNs. This is particularly important on many NLP tasks when a large corpus is involved in training.

5 FOFE-net for KBQA

In this chapter, we present our proposed model, FOFE-net, for the KBQA task as well as its subtasks.

5.1 General Pipeline of Our KBQA system

We tackle the KB-QA task with the following four steps: topic subject discovery, entity linking, relation detection, and constraint detection. In this work, we propose to apply the above FOFE-net model to solve each of the sub-tasks in the pipeline. Given a question, a FOFE based local detector will examine all word segments in a sentence and generates all candidates along with a probability of being a topic subject mention for the question. After pruning, topic subject mention candidates will be used as keywords to look up the Freebase to generate some candidates of entity nodes in Freebase. Each entity node candidate will be scored by an entity linker, which is a FOFE-net model taking each Freebase node and the corresponding subject mention contexts in the question as input to compute the similarity between them. The results from entity linker are fed to the relation detector, which is implemented as another FOFE-net model to score the similarity between the question and the predicates [1] (and paths) from each target Freebase node. Different from the previous works that combine various models in different stages of KB-QA task, our system is mainly built by using the FOFE-net framework. The details for each component in our KB-QA system are described below.

5.2 Topic Subject Mention Discovery

We follow the idea of named entity recognition (NER) in [37] to discover the potential topic subject mentions from each question in the KB-QA task. For each fragment hypothesis and its left and right contexts, FOFE-NER extracts the following features as inputs.

Features for each fragment hypothesis:

- Bag-of-word (BoW) of the underlying fragment, e.g. the bag-of-words vector of 'robert', 'lamm' in Figure 5.1.
- Character-level FOFE code for the underlying fragment viewed as a left-to-right character sequence, e.g. FOFE code of the sequence "*'r', 'o', ..., 'm' , 'm'* " in Figure 5.1.

[1]In Freebase, relations between nodes are labeled with certain predicates.

- Character-level FOFE code for the underlying fragment viewed as a right-to-left character sequence, e.g. FOFE code of the sequence "*'m', 'm', ..., 'o' , 'r'* " in Figure 5.1.
- Character-level CNN representation for the underlying fragment viewed as a character sequence.

Features for contexts of each fragment hypothesis:

- Word-level FOFE codes for the left context, e.g. one FOFE code of the left context including the fragment:"*which songs have robert lamm*", and another FOFE code of the left context excluding the fragment: "*which songs have*".
- Word-level FOFE codes for the right context, e.g. one FOFE code of the right context (including the fragment) as a backward word sequence of "*? to lyrics written lamm robert*", and another FOFE code for the right context excluding the fragment: "*? to lyrics written*".

Different from regular named entity recognition tasks, detected candidates of topic subject mention has to match any names, alias, or Wikipedia keys in freebase. Those failed to match will be discarded. We then rank the remaining candidates based on their probabilities of being a topic subject mention, as computed by the FOFE-net model. Among all candidates, we only keep the ones whose probability

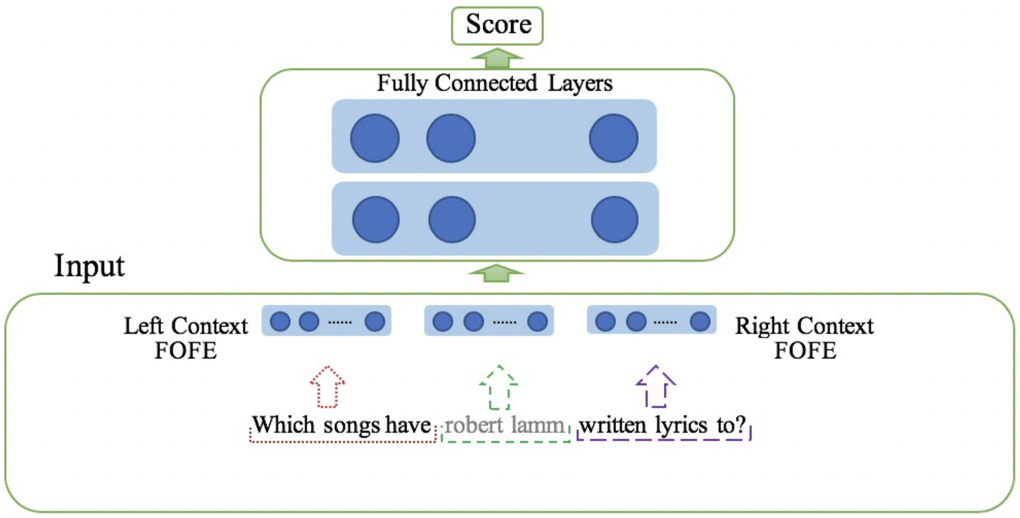


Figure 5.1: The FOFE-net model for NER in the KB-QA pipeline.

is larger than a threshold θ $(0 \leq \theta \leq 1)$.

5.3 Entity Linking

Once a topic subject mention is detected in a question, it will be used as a keyword to look up entity nodes in Freebase. Usually, each detected subject mention will have multiple matched entity nodes in Freebase (See Figure 2.2, subject mention: *Philippines* has three matched entities: $\{m.05v8c,\ m.0ck9jp8,\ m.07ly87t\}$). Thus, an entity linker plays an important role in the process of choosing the correct entity that is relevant to the underlying question. Here, we have built the FOFE-net model for entity linking to determine which entity in Freebase should be selected

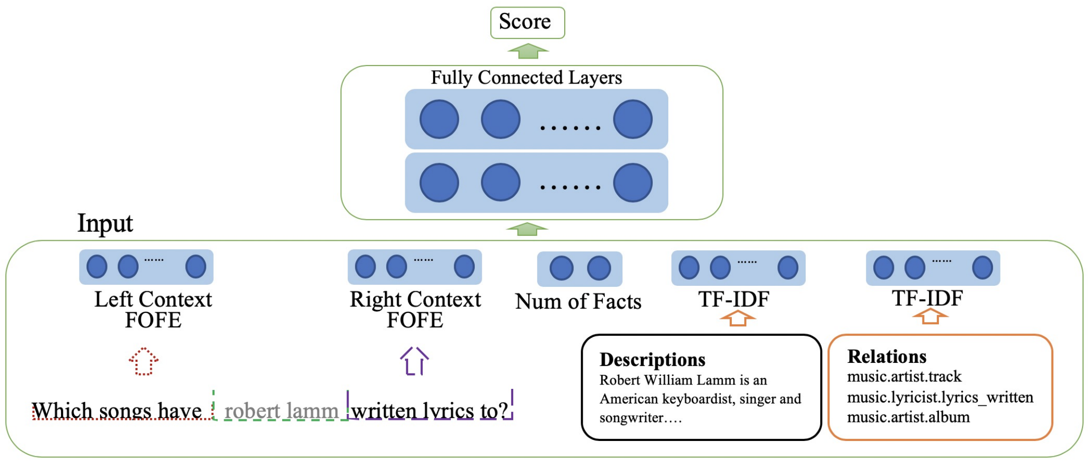


Figure 5.2: The FOFE-net model for entity linking in the KB-QA pipeline. The detected subject mention in this case is "*robert lamm*".

for each detected topic subject mention. To model the relevance between a detected subject mention and each Freebase entity, the FOFE-net model takes both Freebase node and the context of detected subject mention in the question as input. In the following, we describe the features used to represent each input.

Features for each subject in the question: Without losing information, each detected subject mention must be represented by its contexts (both left and right) in the given question. Since its left and right contexts may be viewed as a sequence of words, they may be easily encoded as fixed-size FOFE codes.

As shown by the example in Figure 5.2, the FOFE-net model for entity linking takes the following features for each subject mention in the question:

- Bag-of-word (BoW) of each detected subject, e.g. the bag-of-word vector of 'robert', 'lamm'.

- Word-level FOFE codes for left context, e.g. one FOFE code of left context including the detected subject mention:"*which songs have robert lamm*", and another FOFE code of left context excluding the subject mention: "*which songs have*".

- Word-level FOFE codes for right context, e.g. one FOFE code of right context (including the subject mention) as a backward word sequence of "*? to lyrics written lamm robert*", and another FOFE code for right context excluding the subject mention: "*? to lyrics written*".

Features for each Freebase entity node: The FOFE-net also needs to use features to fully depict the information of entity node in Freebase. In this work, As shown in Figure 5.2, we use the following feature for each Freebase node:

- Number of facts: the total number of associated facts is computed for each node and A hot value is quantized into 10 discrete values and represented as a 10-D one-hot vector [23].

- TF-IDF[2] code for the description of entity node: many Freebase nodes contain a short description of the corresponding entity and a bag of TF-IDF code is

[2]We compute the TF-IDF by using TfidfTransformer provided by sklearn. `https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html`

computed for the short description.

- TF-IDF code for the relations associated with the entity node: the predicates of all associated relations from the Freebase nodes are all viewed as word sequences and another bag of TF-IDF code is computed for them.

Loss Function: We use a hinge loss function to train our FOFE-net model for entity linking:

$$L_{linker} = \max\left(0, \gamma + s_l\left(q_c, e^-\right) - s_l\left(q_c, e^+\right)\right) \tag{5.1}$$

where q_c denotes question contexts for the detected subject mention, e^+ denotes the gold subject entity (a positive samples), and e^- for all other candidate subject entities (negative samples), γ is a margin constant.

Entity Re-Ranking: To further improve the entity linking, we do the entity re-ranking by combining the linking score with NER score and relation detection score:

$$\mathbf{s}\left(q, e\right) = s\left(q, m\right) + s_l\left(q_c, e\right) \quad + \max_{r \in R_e} s_r\left(q, r\right) \tag{5.2}$$

where $s\left(q, m\right)$ is a similarity score computed by the named entity detector between given question and a topic entity mention. $s_l\left(q_c, e\right)$ is a similarity score computed by the entity linker between given question contexts and an entity. $s_r\left(q, r\right)$ is another score computed by the relation detector between a raw question and a relation. R_e denotes the set of all relations associated with entity e. The number of associated

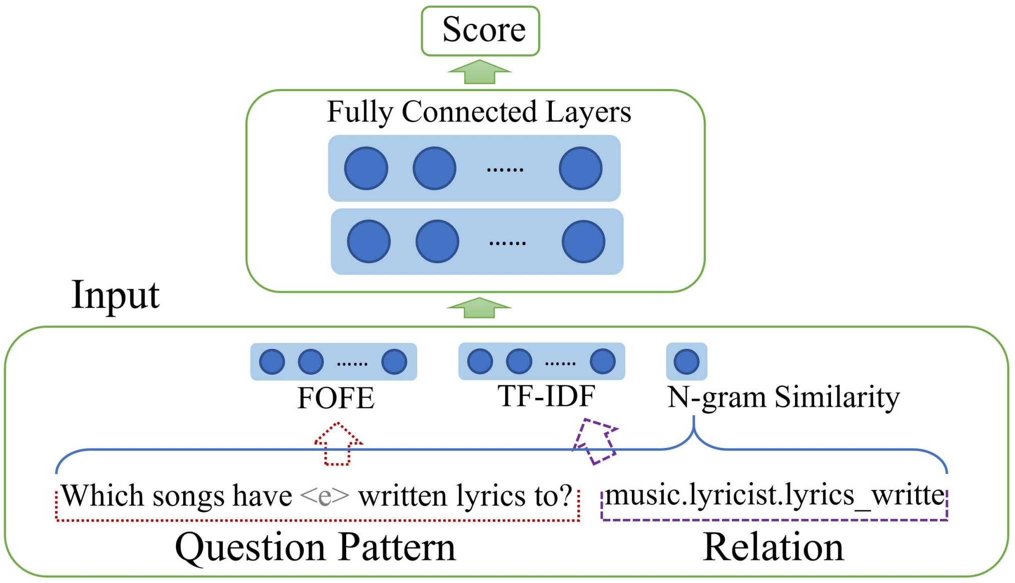


Figure 5.3: The FOFE model for Relation Detection

facts of an entity is used to break the tie when more than one entity has the same score.

5.4 Relation Detection

The relation detection module attempts to detect the correct relation that a question refers to. Here we also use the FOFE-net framework to build our relation detector, which takes a question pattern[3] and a relation as input to compute a similarity score between them. As shown in Figure 5.3, the following features are fed to the FOFE-net model for relation detection:

[3]Question pattern is formatted by replacing the topic subject mention of the question with a special symbol <e>.

Features for Question:

- Word-level FOFE code for the underlying question pattern viewed as a left-to-right word sequence, e.g. FOFE code of the sequence "*which songs have* <e> written lyrics to ?" in Figure 5.3.
- Word-level FOFE code for the question pattern viewed as a right-to-left word sequence, e.g. FOFE code of the sequence "*? to lyrics written* <e> have songs which" in Figure 5.3.

Features for Relation:

- A bag of TF-IDF code for relation when the relation predicates are simply viewed as a bag of words, e.g. the relation of "*music.lyricist.lyrics_written*" is viewed as a bag of words of "*'music', 'lyricist', 'lyrics', 'written'*".
- Character n-gram feature: the overlap of character n-grams between the question and the relation predicate.

Loss Function: Similarly, we also use a hinge loss function to train our FOFE-net model for relation detection.

$$L_{rel} = \max\left(0, \gamma + s_r\left(q_p, r^-\right) - s_r\left(q_p, r^+\right)\right) \tag{5.3}$$

where q_p denotes the underlying question pattern, r^+ for its gold relation (as positive example), and r^- for other candidate relations associated with the node (as negative samples), and γ denotes a constant margin parameter.

5.5 Answer Selection

Answer selection has two main steps: path scoring and answer scoring. For the path scoring step, we score each candidate paths (m,e,r) based on the following formula:

$$s\left(m, e, r; q\right) = \mathbf{s}\left(q, e\right) + s_r\left(q_p, r\right) \tag{5.4}$$

With the score of each path, we can score each answer candidate based on the following formula:

$$s_a = \sum_{(m,e,r)\in\left(P_q^N \cap P_a\right)} s\left(m, e, r; q\right) \tag{5.5}$$

where P_a is the paths that can lead to the answer a, P_q^N is the top N best scoring paths of the question q.

This step is called answer scoring. In this step, the top N best scoring paths of the question q are selected to generate the candidate answers to the question. The score of each candidate answer is the sum of the scores of the paths that can lead to it. We keep the candidate answers with the highest scores as the final candidate answer set.

5.6 Constraint Detection

In WebQSP dataset and FreebaseQA, many of candidate answers of the question can be queried by the same subject-relation pair (which means they may have the same score), while only a few of them are correct because of various kind of constraints implied by the question. Similar to [40, 41, 43], we also use a few pre-defined keywords to detect temporal constraints and ordinal constraints[4]. The type entity can be suggested by the predicted inferential chain and detected through text matching.[5] For example, inferential chain:{*Governing Official, Government Position Holder*} suggests several possible constraints: *President*, *Vice President* and *Prime Minister*, etc. Through text matching we can see *President* is the best match constraint to the question: *"who was elected president of the Philippines"*. All detected constraints will be used to prune the answer candidates and produce the final answers to the question.

[4]For example, {was, were, did} and {first, last} is used to detect past constraint and ordinal constraint in the question respectively.

[5]We have collected all inferential chains and their associated constraints from the training set.

6 FreebaseQA: A New Factoid QA Dataset

A number of datasets are available for KBQA, such as Free917 [9], WebQuestions [3], WebQuestionsSP (WebQSP) [41], SimpleQuestions [7], ComplexQuestions [1]. However, these existing datasets are either small-sized or linguistically simple. In this chapter, we first discuss three of popular KBQA benchmarks and then elaborate our method to create a new KBQA dataset, FreebaseQA [19].

6.1 KBQA Benchmarks

6.1.1 WebQuestions

WebQuestions is introduced in [3], which contains 5,810 question-answer pairs. The questions in this dataset are obtained by Google Suggest API and involve human judgment to ensure these questions can be answered over Freebase. Since their question generation procedure is completely independent from the Freebase, questions in this dataset are more natural and sensible. Besides, the statistics also show this dataset is very diverse lexically.

6.1.2 WebQuestionsSP

WebQuestionsSP (WebQSP) [41] is another dataset from the family of WebQuestions, which enhances the WebQuestions with human-annotated semantic parses and questions removal for those that are not answerable by using Freebase. Their method results in 3,098 questions for training and 1,639 questions for validation. Moreover, they empirically demonstrate that training with semantic parse labels can lead to the improvement of KBQA system.

6.1.3 SimpleQuestions

SimpleQuestions [7] consists of a total of 108,442 questions written by human annotators with corresponding facts extracted from Freebase. Compared to WebQuestions, this dataset has a larger scale but has obvious drawbacks. First, the generation of question is highly dependent on Freebase, which asks human annotators to phrase the question according to given Freebase facts, albeit ambiguous and unlikely to be asked in real life. This results in the appearance of many unnatural questions, such as "what is the book e about" and "what country is fearless from".[6] Second, all questions are generated based on a single fact, which makes it less challenging for modern KBQA system.

[6]In the second example, fearless refers to an album released in 1998.

6.2 Constructing the FreebaseQA Dataset

In light of many shortcomings of existing KBQA benchmarks discussed above, we build a new large-scale factoid QA dataset, FreebaseQA, which consists of 54,611 matches and 28,348 unique questions. Similar to SimpleQuestions, each question in FreebaseQA is paired with answers and corresponding facts, which makes the training and evaluation on KBQA subtasks, i.e. named entity recognition (NER), entity linking (EL) and relation detection (RD), possible.

6.3 Preparation of Question-Answer Pairs

In FreebaseQA, rather than generate any new question-answer pairs, we have collected pre-composed trivia-type factoid questions from various sources. Different from SimpleQuestions and WebQSP, these questions are independently composed for human contestants in trivia-like competitions. As a result, these questions are more complicated linguistically and have a broader coverage for diverse topics than almost all existing KBQA datasets.

Particularly, we prepare our QA pairs by choosing the TriviaQA [20] dataset as the primary source with the supplement of questions collected from the trivia websites, KnowQuiz (`http://www.knowquiz.com`), QuizBalls(`http://www.quizba-lls.com`), and QuizZone (`https://www.quiz-zone.co.uk`). Duplicate entries are removed and the rest of them are merged into a single source. We then run two named entity recognition (NER) systems: TAGME [12] and FOFE NER [37] on each question for the extraction of possible subjects. Confidence thresholds of 0.2 and 0.9 are chosen for the respective systems to ensure the quantity and the quality of the produced subjects.

6.4 Freebase Matching

The matching step starts with Freebase nodes corresponding to the produced subjects. For each such node, we retrieve object nodes that directly connect to it, and then check if the names or the alias of retrieved object nodes can match the answer to the question. Once a match is found, Freebase ID of the subject node, the predicate, and the Freebase ID of the object node are saved as a supporting triple the represents the question-answer pair. Note that one question-answer pair may have multiple triple matches.

However, this matching procedure is inefficient as it only goes unidirectionally, i.e. from the subject nodes in the question to the answer nodes. For the optimization, we adopt a two-way search strategy, which enables the matching procedure to start from both subject and answer nodes. Then, the search concludes when the same object node is found from both starting points of the search. By using this search strategy, we have accelerated the Freebase matching algorithm by more

Components	**Example 1**	**Example 2**
Question [Answer]	`Which 18th century author wrote Clarissa (or The History of a Young Lady), said to be the longest novel in the English language? [Samuel Richardson]`	`What is the correct name of the character voiced by Angela Lansbury in Beauty and The Beast? [Mrs Potts]`
Subject (Freebase ID)	`Clarissa (m.05s1st)`	`Angela Lansbury (m.0161h5)`
Predicate	`book.written-work.author`	`film.actor.dubbing-performances`
Secondary Predicate	-	`film.dubbing-performance.character`
Object/Answer (ID)	`Samuel Richardson (m.0hb27)`	`Mrs Potts (m.02vw823)`

Table 6.1: Two typical examples to illustrate the data format of all matches in FreebaseQA.

than ten-fold.

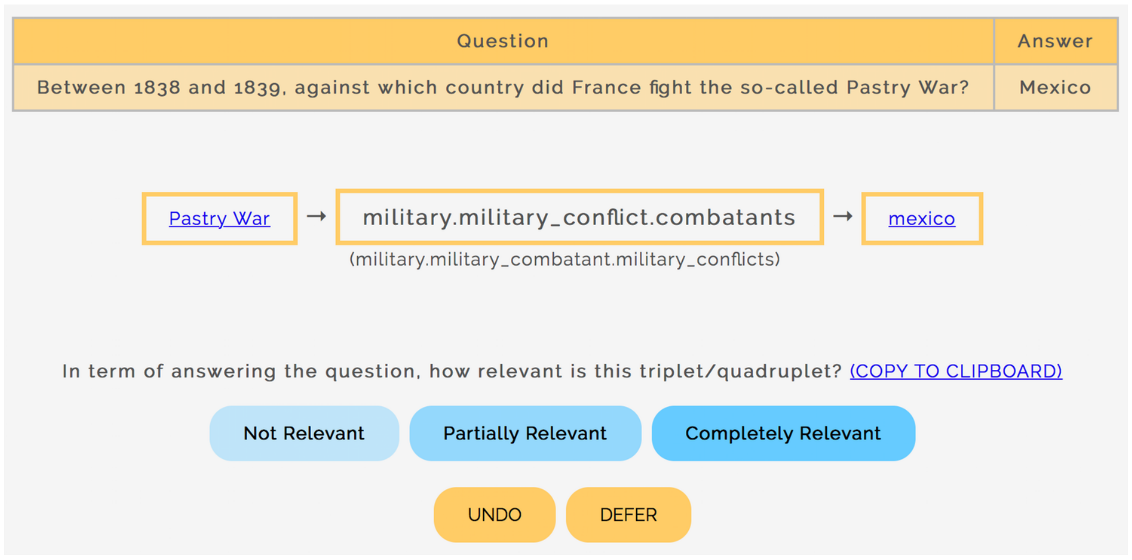
6.5 Mediator Nodes

Freebase is curated to have special type of nodes called a mediator node. A mediator is an intermediate node that connects a subject node to an object node through a two-hop predicates. These mediator nodes do not have any name or alias associated with them, but they are crucial when constructing the FreebaseQA dataset. In the FreebaseQA dataset, we assume that all answers are reachable from the subject entities in the question through either paths of length 1 or paths of length 2 via the mediator nodes. If the above two-way search reaches a mediator, our searcher will go one step further to look through all nodes linked to this mediator to acquire a secondary predicate so as to bridges the subject to the answer via the mediator node. We provide an example to illustrate how the mediator mechanism works, which is described as Example 2 in Table 6.1.

6.6 Human Annotation

To assure the high-quality supervision, the human verification of the collected matches is involved. We hire 10 native English speakers for the annotation of all the collected matches. Each human annotator is asked to rate the level of relevancy of the triple matches to corresponding questions. We have defined three levels of relevancy, "Completely Relevant", "Somewhat Relevant" and "Not Relevant". Each individual only needs to make a one-out-of-three choice throughout the annotation process, which make the human involvement in the creation of FreebaseQA dataset very light. Unlike the creation of SimpleQuestions [7] and ComplexWebQuestions [34] in which human annotators are asked to compose or rephrase the questions, this method significantly reduce the cost of QA data collection. Fig. 6.1 illustrates the user interface for this data annotation procedure.

In order to facilitate the model training, the matches rated "Completely Rele-



Question	Answer
Between 1838 and 1839, against which country did France fight the so-called Pastry War?	Mexico

Pastry War → military.military_conflict.combatants → mexico

(military.military_combatant.military_conflicts)

In term of answering the question, how relevant is this triplet/quadruplet? (COPY TO CLIPBOARD)

Figure 6.1: Human annotators use this website interface to label all automatically-generated matches, rating either Completely Relevant, Partially Relevant, or Not Relevant.

vant" by human annotators are randomly chosen to populate the training, evaluation, and development sets of FreebaseQA, respectively. We also ensure that all of such matches for a single question-answer pair can only exist in one of the sets. Moreover, the matches rated Partially Relevant are kept as a separate set, which may be useful for the model training as well.

A summary of the number of collected question-answer pairs along with the number of matches from each source are provided in Table 6.2. We see that except for KnowQuiz, the number of matches in Freebase roughly equals to the number of questions in each source. Among all the generated matches, 54,611 matches in total are kept as true positives by human annotators.

Source	**Questions**	**Matches**
TriviaQA	98,973	99,523
KnowQuiz	9,996	2,389
QuizBalls	15,370	17,856
QuizZone	7,686	7,289
Total	132,025	127,057

Table 6.2: A summary of the number of question-answer pairs from each source along with the number of matches generated from the above Freebase matching procedure.

Since multiple Freebase matches may refer to the same collected question-answer pair, we then merge the matches referring to same question-answer pair together, which result in a new QA dataset consisted of 28k unique questions. We name this new dataset as FreebaseQA and release it for public use. [7]

[7]`https://github.com/infinitecold/FreebaseQA`

6.7 Statistics of FreebaseQA

Table 6.3 and 6.4 provides some Statistics of FreebaseQA as well as that of two popular QA datasets, SimpleQuestions and WebQSP. We can see that FreebaseQA is larger than WebQSP in the number of unique questions while it is only a quarter of SimpleQuestions in the number of unique questions.

Dataset	train	dev	eval	**Total**
SimpleQuestions	75,910	10,845	21,687	108,442
WebQSP	3,098	-	1,639	4,737
FreebaseQA	20,358	3,994	3,996	28,348

Table 6.3: Total numbers of unique questions found in each data set.

Dataset	The Average # of Words Per Question
SimpleQuestions	7.65
WebQSP	6.62
FreebaseQA	13.35

Table 6.4: The average number of words per question on each data set.

Another important factor that should be taken into account for these datasets is

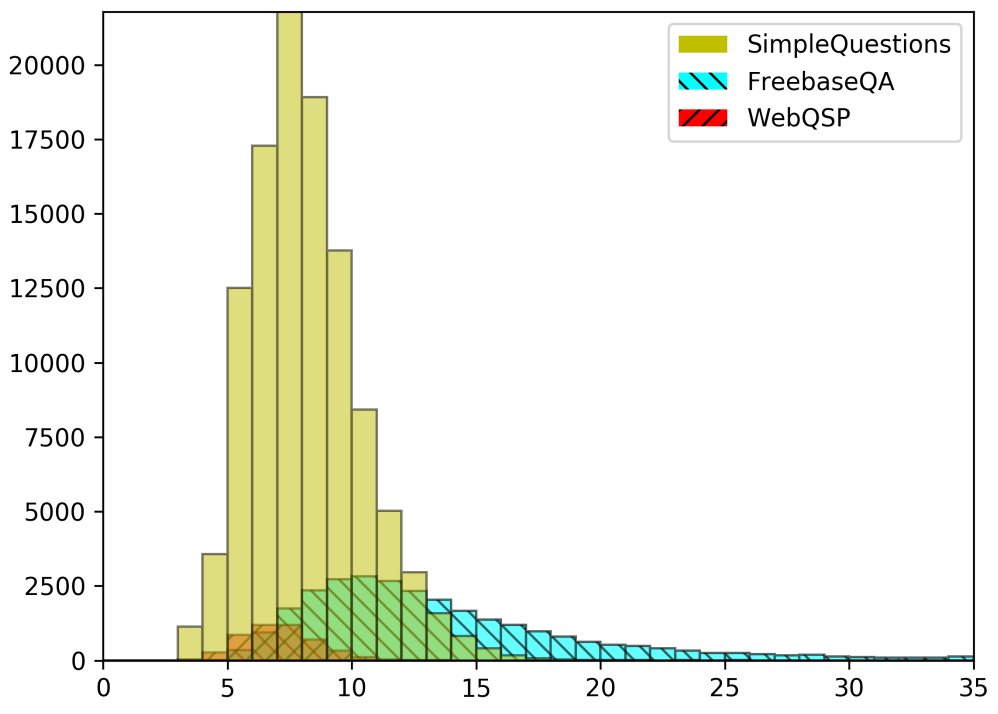


Figure 6.2: A histogram showing the spread of the length of the questions in each data set.

their linguistic sophistication. We use question length, i.e. the number of words, as one of the metric to evaluate how linguistic sophisticated the questions are on each dataset. From Table 6.4, we see that the average question length on FreebaseQA is 13.81, about double the length for either dataset. We also present a histogram to depict the spread of the length of the questions in each data set, which is shown in Fig. 6.2.

We provide three sample questions in Table 6.5 to prove that FreebaseQA is more difficult. These three questions are asking for the same predicate, *location.partially_contains*. It is quite obvious that questions in FreebaseQA is not only longer but also more diverse in language use. Besides, long questions are more likely to contain entities (German, North Sea and Hamburg in the example question) that does not directly connect to the answer. This may make entity linking process more challenging since it is not trivial for entity linker to know the Czech Republic is the topic entity instead of German or North sea or Hamburg.

Dataset	SQ	WQ	FQ
Questions	Which river is partially located in Belgium?	What is the rainforest in Peru called?	Which German river flows 1159 kilometres from the Czech Republic flowing into the North Sea just north of Hamburg?
Subject	Belgium (m.0154j)	Peru (m.016wzw)	Czech Republic (m.01mjq)
Predicate	location.partially contains	location.partially contains	location.partially contains
Secondary Predicate	-	-	-
Object/Answer(ID)	Sambre (m.03xhvv)	Andes (m.0p2n)	Elbe (m.0ddlt)

Table 6.5: Sample questions extracted from SimpleQuestions (SQ), WebQSP (WQ) and FreebaseQA (FQ) respectively. The red text refers to topic entities that connect to the answer, the blue text refers to other entities.

7 Experiments

To evaluate the effectiveness of our proposed FOFE-net models, we have used two popular KB-QA dataset, SimpleQuestions, WebQSP and our newly created dataset, FreebaseQA, in our experiments. We compare our overall QA performance with other systems reported in the published literature [40, 41, 42, 43, 26, 15, 33]. Meanwhile, we also investigate the performance of each FOFE-based module in the KB-QA pipeline, i.e. entity linking and relation detection, and compare with other results reported under the same experimental settings [42, 43, 29].

7.1 Datasets and Evaluation Metric

SimpleQuestions: The SimpleQuestions dataset [7] is a single-relation KB-QA dataset provided by Facebook along with two extracted subsets of Freebase, i.e. FB2M, and FB5M. Following previous works[7, 42, 43, 26, 15], we use FB2M as the knowledge base and evaluate our system in terms of subject-relation pair accuracy on this benchmark.

WebQSP: WebQSP dataset is a small-scale KB-QA dataset that contains single-relation and multi-relation questions [41]. Following previous works [41, 43], we use entire Freebase for this dataset and choose true accuracy, i.e., a question is considered answered correctly only if the predicted answer set is exactly same as one of the answer sets, as the evaluation metric. We use the official evaluation script to evaluate our system on this benchmark.

FreebaseQA: FreebaseQA dataset [19] is a large-scale dataset with 28K unique open-domain factoid questions which collected from triviaQA dataset [20] and other trivia websites (KnowQuiz : `http://www.knowquiz.com`, QuizBalls : `http://www.quizballs.com`, QuizZone : `https://www.quiz-zone.co.uk`). All of the questions in this dataset have been matched to the Freebase for generating the subject-predicate-object triples and verified by human annotators. We generate a new extract of Freebase (including 182M triples, 16M unique entities) for this dataset and choose true accuracy as the evaluation metric.[8]

7.2 Experimental Configurations

We use the same settings as that of [37] for the topic subject detection model. For the entity linking and relation detecting models, our settings are described as

[8]The Freebase extract can be found in: `https://www.dropbox.com/sh/dkg02j3uwehkt1j/AADqofTAPRA7QpKkhPSW-CJva?dl=0`

below. Our hyperparameters are chosen empirically based on the performance of the development sets.

Network structure: For the feedforward neural networks for entity linking, we choose a structure of 4 fully-connected layers of 1024 nodes with ReLu activation functions on all the datasets. For the network structure for the relation detection model, we use a network of 4 fully-connected layers with ReLu activation function on all the datasets. The number of node in each layer for SimpleQuestions, WebQSP, FreebaseQA is 735, 256, 600 respectively.

Embedding Matrices: For the entity linker, we use case-insensitive pre-trained word embeddings of 128 dimensions on all the datasets. For our relation detector, we choose case-insensitive pre-trained word embeddings with 128 dimensions on WebQSP and FreebaseQA, and case-insensitive pre-trained word embeddings with 256 dimensions on SimpleQuestions. Our vocabulary is constituted of the top 100k frequent words from English gigaword [27].

Dropout Rate: For relation detection models, we have set the dropout rate to 0.24, 0.05, 0.15 on SimpleQuestions, WebQSP and FreebaseQA datasets respectively. For entity linking, we set the dropout rate to 0.15 for all the datasets.

Optimizer and Learning rate: To minimize the loss function of our models, we use the standard SGD (stochastic gradient descent) optimizer with a slow schedule to decay the learning rate. The learning rate is initialized to be 0.01.

Forgetting Factor: we have used $\alpha = 0.8$ for relation detection on all datasets, and $\alpha = 0.95$, $\alpha = 0.95$ and $\alpha = 0.8$ for entity linking on SimpleQuestions, WebQSP and FreebaseQA datasets respectively.

Training Data Preparation: To train our FOFE-net models for topic subject discovery, entity linking and relation detection on each specific QA dataset, we only use entire Freebase or its extract to generate negative samples based on this specific QA dataset for model training.

Our proposed model is compared with the following baseline methods:

- STAGG [41]: Staged Query Graph Generation (STAGG) algorithm views a KBQA task as a staged problem, so they adopt Structured Multiple Additive Regression Trees (S-Mart) model for entity linking and deep CNN for identifying the relation.

- AMPCNN [42]: This method uses bidirectional LSTM with Conditional Random Field (CRF) model for entity mention detection and Attentive MaxPooling CNN (AMPCNN) for predicate-pattern matching.

- HR-BiLSTM [43]: Hierarchical Bidirectional LSTM (HR-BiLSTM) is focused on relation detection task. The author combines this model with S-Mart and AMPCNN entity linker for the WebQSP dataset and SimpleQuestions dataset respectively.

- BiLSTM-BiGRU [26]: Author builds a strong baseline by using Bidirectional LSTM as an entity linker and Bidirectional GRU for relation detection.

- PESQA [15]: Pattern-revising Enhanced Simple Question Answering (PESQA) utilizes BiLSTM-CRF model to detect the entity mention span and sophisticated heuristics to improve the pattern extraction. LSTM is adopted as a relation detector in their settings.

- GRAFT-Net [33]: This method builds a sub-graph of KB by running an existing entity linker, S-Mart, on the question. The edges of such sub-graph are weighted by the similarity of their corresponding relation vector to the LSTM encoding of the question. The Personalized PageRank algorithm is used to propagate the embeddings through the graph and the answers are selected by doing binary classification on each candidate nodes.

7.3 Overall KB-QA Performance

In Table 7.1, we have listed the overall KB-QA performance of our models on two datasets and compared with other stage-based models reported in the published literature. Column 2 and column 3 list the models that they used for entity discovery and linking (EDL) and relation detection (RD). We can see the previous works usually combine various models together for the KB-QA task, while our work only

Previous works	EDL	RD	Accuracy	
			SQ	WQ
STAGG [41]	S-MART	CNN	-	63.9
AMPCNN [42]	BiLSTM-CRF	AMPCNN	76.4	-
HR-BiLSTM [43]	BiLSTM-CRF/S-MART	HR-BiLSTM	77.0	63.0
BiLSTM-BiGRU [26]	BiLSTM	BiGRU	75.1	-
PESQA [15]	BiLSTM-CRF	LSTM	**80.2**	-
GRAFT-Net [33]	S-MART	LSTM	-	66.7
Our work	FOFE-net	FOFE-net	77.3	**67.6**

Table 7.1: Comparison of the overall KB-QA accuracy (in %) on the test sets of SimpleQuestions (SQ) and WebQSP (WQ).

Models	Accuracy		
	SQ	WQ	FQ
HR-BiLSTM [43]	77.0	63.0	-
FOFE-net	77.3	67.6	**37.0**

Table 7.2: Accuracy (in %) on the test (eval) sets of SimpleQuestions (SQ), WebQSP (WQ) and FreebaseQA (FQ).

use FOFE-net throughout all the stages, which makes the pipeline easy to implement. Our FOFE-net also achieves very strong performance on the two examined data sets. As shown, our FOFE-net based model is competitive with other CNN, LSTM, regression tree based state-of-the-art model combinations on SimpleQuestions benchmark, and outperform GRAFT-Net [33] by 0.9% on WebQSP in terms of true accuracy. These results prove the effectiveness of our proposed model and suggest that the FOFE-net can be universally applied to other NLP tasks, such as entity discovery and linking (EDL) and relation detection.

Hao et al. [15] report a fresh result on SimpleQuestions dataset that outperforms other previous works as well as ours by a quite large margin (around 3%). Their model is enhanced by a quite sophisticated rule-based pattern revising method that pushes their performance from 77.8% to 80.2%. Compared to their model, we only use a few heuristics and simple functions in combining models at different stages. This simplicity may suffer from some disadvantages on performance while we believe it can make the pipeline more straightforward and easier to adapt to other KB-QA tasks.

Table 7.2 shows that our FOFE-net model achieves 37.0% in terms of accuracy on FreebaseQA dataset. This performance is much worse than that on other datasets, which prove that our FreebaseQA dataset is more challenging than SimpleQuestions and WebQSP.

Top-K	**1**	**10**	**20**	**50**
AMPCNN [42]	72.7	86.9	88.4	90.2
HR-BiLSTM [43]	79.0	89.5	90.9	92.5
HTTED [29]	81.1	91.7	93.4	**95.1**
Our FOFE-net	**82.2**	**92.5**	**93.6**	94.7

(a) The accuracy (in %) comparison of various entity linking models on the test set of SimpleQuestions in the number of candidates kept in the list.

Top-K	**1**	**2**	**3**	**5**
S-Mart [39]	88.0	**93.3**	**94.7**	**96.0**
Our FOFE-net	**89.1**	93.2	93.7	94.1

(b) The accuracy (in %) comparison of various entity linking models on the test set of WebQSP in the number of candidates kept in the list.

Top-K	**1**	**2**	**3**	**5**	**10**	**20**	**50**
Our FOFE-net	52.4	70.7	79.6	85.7	89.3	90.8	92.0

(c) The accuracy (in %) of FOFE-net on the test set of FreebaseQA in the number of candidates kept in the list.

Table 7.3: Entity Linking Result of FOFE-net on test (eval) set of SimpleQuestions, WebQSP and FreebaseQA.

	Accuracy		
Models	**SQ**	**WQ**	**FQ**
AMPCNN [42]	91.3	-	-
HR-BiLSTM [43]	93.3	82.53	-
Our FOFE-net	**93.3**	**83.26**	76.6

Table 7.4: Comparison of relation detection accuracy (in %) on the test sets of SimpleQuestions (SQ), WebQSP (WQ) and FreebaseQA (FQ).

7.4 Entity Linking Results

To investigate the performance of FOFE-net in entity linking task, we have compared it with three state-of-the-art entity linkers [42, 43, 29] on SimpleQuestions and S-Mart [39] entity linker on WebQSP benchmark. From the results in Table 7.3a and Table 7.3b, we can see that our entity linker achieves a very competitive result in this subtask and outperforms the Hierarchical Types constrained Topic Entity Detection (HTTED) model in terms of top 1 accuracy.[9]

According to the results in Table 7.3a, 7.3b, and 7.3c, we can also see that entity linking on FreebaseQA is more difficult than it is on WebQSP and SimpleQuestions. To analyze the reason, we calculate the average ratio of the number of topic entity

[9]In sec. 5.2, we choose a large θ to prevent entity linker from being overwhelmed by a large number of candidate entities, which affects the entity linking result when K goes large.

Model	Word-level Features	Char-level Feature	Accuracy
Our FOFE-net	FOFE(Q)+TF-IDF(R)	✓	**83.26**
	FOFE(Q)+TF-IDF(R)	✗	81.44
	BOW(Q)+TF-IDF(R)	✓	80.29
	FOFE(Q)+BoW(R)	✓	81.30

Table 7.5: The relation detection accuracy (in %) comparison of various features combination of FOFE-net on the test set of WebQSP

mention candidates to the number of gold topic entity mentions for questions on test (eval) set of three datasets. The results show that that ratio on SimpleQuestions, WebQSP and FreebaseQA is 8.0, 4.4, 11.6 respectively, which have a linear positive correlation with the average length of questions in these datasets. Same as we discussed in Chapter 6, too many redundant matches in Freebase may confuse the entity linker and worsen its performance.

7.5 Relation Detection Results

To investigate the FOFE-net performance on relation detection task, we have compared it with the best relation detector reported in the published literature [42, 43] on SimpleQuestions and WebQSP benchmark. As shown in Table 7.4, our FOFE-net based relation detector yields very competitive results on both dataset, which ties HR-BiLSTM on SimpleQuestions and slightly outperform it on WebQSP by 0.73%.

From Table 7.4, we can also see that even the result our FOFE-net model achieves on FreebaseQA is quite reasonable, the number is still worse than it is on either WebQSP or SimpleQuestions dataset. This is another proof of the value of our FreebaseQA dataset, since it poses a bigger challenge to the KBQA system.

To figure out what contributes more to the FOFE-net on relation detection, we conduct an ablation test. Result is shown in Table 7.5. The best result was achieved by using FOFE code of question, TF-IDF code of relation, and char-level ngram as features. By removing the char-level feature, the result drops from 83.26% to 81.44%. This proves char-level feature is important on relation detection because it can alleviate the zero-shot problem, i.e., a relation is unseen in the training set but similar to the question in the surface form. By replacing FOFE code of question with a sum of Bag-of-word (BoW), the performance drops by around 3%. This proves FOFE successfully captures the useful ordinal information in the question and improves the model's performance on the relation detection task. By replacing TF-IDF code of relation with a sum of Bag-of-word (BoW), the performance drops by 1.96%, we believe it is because some relations share similar lexicons, e.g. location.geocode.latitude and location.geocode.longitude, a sum of

Bag-of-word (BoW) is hard to represent these relations distinctively. However, TF-IDF can do it well by assigning a low value to the frequent words, i.e. location and geocode, and high value to the infrequent words, i.e. latitude and longitude.

8 Conclusion

In this chapter, we will give our conclusion of this thesis and discuss the potential follow-up works.

8.1 Conclusion

FOFE, a simple fixed-size encoding method, has been successfully applied to many NLP tasks. In this thesis, we have further extended this method to solve several sub-tasks in the KB-QA pipeline, including topic subject discovery, entity linking and relation detection. Our experimental results have shown that this simple framework, FOFE-net, works well in all examined sub-tasks, and in turn pushing our overall KBQA system to achieve competitive results on popular KBQA datasets.

Besides, we present a new data set, FreebaseQA, for open-domain factoid QA over structured knowledge bases in this thesis. FreebaseQA is shown to be more challenging than currently available KBQA datasets. Therefore, FreebaseQA may be a valuable resource for further investigation of more advanced KBQA system.

8.2 Future Works

There are several possible improvements, which could be done in the future. We divide them into two directions:

Compositional Questions Currently FreebaseQA assumes that all questions can be answered via one-hop from the topic entity. This assumption is somehow too naive to test the machine as a human being is capable of answering the question through multiple steps of reasoning. Many emergent reading comprehension tasks have already involved this type of question, which can be one of the sources to extract more data for the expansion of our FreebaseQA dataset.

Entity Linking Our entity linker extracts the topic entity from the question solely based on the textual information of itself as well as that of its context. However, the interaction between the topic entity and other entities conveyed in the question should be taken into account. Take the sample question in Table 6.5 as an example, German, Hamburg and North Sea are assumed to connect to the true topic entity, Czech Republic, on the Freebase. Then how to leverage this as supportive information for entity linking will be left for future exploration.

Bibliography

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

[2] H. Bast and E. Haussmann. More accurate question answering on freebase. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 1431–1440, New York, NY, USA, 2015. ACM.

[3] J. Berant, A. Chou, R. Frostig, and P. Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of EMNLP 2013*, pages 1533–1544. Association for Computational Linguistics, 2013.

[4] 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), pages 1415–1425. Association for Computational Linguistics, 2014.

[5] 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*, pages 1247–1250. ACM, 2008.

[6] A. Bordes, S. Chopra, and J. Weston. Question answering with subgraph embeddings. In *Proceedings of the EMNLP 2014*, pages 615–620. Association for Computational Linguistics, 2014.

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

[8] A. Bordes, J. Weston, and N. Usunier. Open question answering with weakly supervised embedding models. In T. Calders, F. Esposito, E. Hüllermeier, and R. Meo, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 165–180, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[9] 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), pages 423–433, Sofia, Bulgaria, Aug. 2013. Association for Computational Linguistics.

[10] Z. Dai, L. Li, and W. Xu. Cfo: Conditional focused neural question answering with large-scale knowledge bases. In *Proceedings of ACL 2016*, pages 800–810. Association for Computational Linguistics, 2016.

[11] M. Fabian, K. Gjergji, W. Gerhard, et al. Yago: A core of semantic knowledge unifying wordnet and wikipedia. In *16th International World Wide Web Conference, WWW*, pages 697–706, 2007.

[12] P. Ferragina and U. Scaiella. Tagme: On-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 1625–1628, New York, NY, USA, 2010. ACM.

[13] Y. Goldberg and G. Hirst. *Neural Network Methods in Natural Language Processing*. Morgan & Claypool Publishers, 2017.

[14] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. The MIT Press, 2016.

[15] Y. Hao, H. Liu, S. He, K. Liu, and J. Zhao. Pattern-revising enhanced simple question answering over knowledge bases. In *Proceedings of COLING 2018*, pages 3272–3282. Association for Computational Linguistics, 2018.

[16] Y. Hao, Y. Zhang, K. Liu, S. He, Z. Liu, H. Wu, and J. Zhao. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In *Proceedings of ACL 2017*, pages 221–231. Association for Computational Linguistics, 2017.

[17] X. He and D. Golub. Character-level question answering with attention. In *Proceedings of EMNLP 2016*, pages 1598–1607. Association for Computational Linguistics, 2016.

[18] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Netw.*, 4(2):251–257, Mar. 1991.

[19] K. Jiang, D. Wu, and H. Jiang. FreebaseQA: A new factoid QA dataset matching trivia-style question-answer pairs with Freebase. In *Proceedings of NAACL-HLT 2019*. Association for Computational Linguistics, 2019.

[20] M. Joshi, E. Choi, D. Weld, and L. Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of ACL 2017*, pages 1601–1611. Association for Computational Linguistics, 2017.

[21] D. Jurafsky and J. H. Martin. *Speech and Language Processing (2Nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2009.

[22] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.

[23] D. Liu, W. Lin, S. Zhang, S. Wei, and H. Jiang. Neural networks models for entity discovery and linking. *CoRR*, abs/1611.03558, 2016.

[24] D. Lukovnikov, A. Fischer, J. Lehmann, and S. Auer. Neural network-based question answering over knowledge graphs on word and character level. In *Proceedings of WWW 2017*, WWW '17, pages 1211–1220, Republic and Canton of Geneva, Switzerland, 2017. International World Wide Web Conferences Steering Committee.

[25] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS 2013*, NIPS'13, pages 3111–3119, USA, 2013. Curran Associates Inc.

[26] S. Mohammed, P. Shi, and J. Lin. Strong baselines for simple question answering over knowledge graphs with and without neural networks. In *Proceedings of*

NAACL-HLT 2018, pages 291–296. Association for Computational Linguistics, 2018.

[27] R. Parker, D. Graff, J. Kong, K. Chen, and K. Maeda. English gigaword. *Linguistic Data Consortium*, 2011.

[28] E. Prudhommeaux and A. Seaborne. Sparql query language for rdf. w3c recommendation, january 2008, 2008.

[29] Y. Qiu, M. Li, Y. Wang, Y. Jia, and X. Jin. Hierarchical type constrained topic entity detection for knowledge base question answering. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, pages 35–36, Republic and Canton of Geneva, Switzerland, 2018. International World Wide Web Conferences Steering Committee.

[30] Y. Qu, J. Liu, L. Kang, Q. Shi, and D. Ye. Question answering over freebase via attentive RNN with similarity matrix based CNN. *CoRR*, abs/1804.03317, 2018.

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

[32] J. Sanu, M. Xu, H. Jiang, and Q. Liu. Word embeddings based on fixed-size ordinally forgetting encoding. In *Proceedings of EMNLP 2017*, pages 310–315. Association for Computational Linguistics, 2017.

[33] H. Sun, B. Dhingra, M. Zaheer, K. Mazaitis, R. Salakhutdinov, and W. Cohen. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of EMNLP 2018*, pages 4231–4242. Association for Computational Linguistics, 2018.

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

[35] S. Watcharawittayakul, M. Xu, and H. Jiang. Dual fixed-size ordinally forgetting encoding (fofe) for competitive neural language models. In *Proceedings of EMNLP 2018*, pages 4725–4730. Association for Computational Linguistics, 2018.

[36] K. Xu, S. Reddy, Y. Feng, S. Huang, and D. Zhao. Question answering on freebase via relation extraction and textual evidence. In *Proceedings of ACL 2016*, pages 2326–2336. Association for Computational Linguistics, 2016.

[37] M. Xu, H. Jiang, and S. Watcharawittayakul. A local detection approach for named entity recognition and mention detection. In *Proceedings of ACL 2017*, pages 1237–1247. Association for Computational Linguistics, 2017.

[38] M. Yahya, K. Berberich, S. Elbassuoni, M. Ramanath, V. Tresp, and G. Weikum. Natural language questions for the web of data. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 379–390. Association for Computational Linguistics, 2012.

[39] Y. Yang and M.-W. Chang. S-mart: Novel tree-based structured learning algorithms applied to tweet entity linking. In *Proceedings of ACL 2015*, pages 504–513. Association for Computational Linguistics, 2015.

[40] S. W.-t. Yih, M.-W. Chang, X. He, and J. Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. ACL – Association for Computational Linguistics, July 2015.

[41] 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 ACL 2016*, Berlin, Germany, August 2016. Association for Computational Linguistics.

[42] W. Yin, M. Yu, B. Xiang, B. Zhou, and H. Schütze. Simple question answering by attentive convolutional neural network. In *Proceedings of COLING 2016*, pages 1746–1756. The COLING 2016 Organizing Committee, 2016.

[43] M. Yu, W. Yin, K. S. Hasan, C. dos Santos, B. Xiang, and B. Zhou. Improved neural relation detection for knowledge base question answering. In *Proceedings of ACL 2017*, pages 571–581. Association for Computational Linguistics, 2017.

[44] S. Zhang, H. Jiang, M. Xu, J. Hou, and L. Dai. The fixed-size ordinally-forgetting encoding method for neural network language models. In *Proceedings of ACL 2015*. Association for Computational Linguistics (ACL), 2015.