Knowledge Graph Embedding Based Question Answering

Xiao Huang, Jingyuan Zhang, Dingcheng Li, Ping Li
Cognitive Computing Lab (CCL), Baidu Research, USA
huangxiao518@gmail.com,{zhangjingyuan03,lidingcheng}@baidu.com,pingli98@gmail.com

ABSTRACT

Question answering over knowledge graph (QA-KG) aims to use facts in the knowledge graph (KG) to answer natural language questions. It helps end users more efficiently and more easily access the substantial and valuable knowledge in the KG, without knowing its data structures. QA-KG is a nontrivial problem since capturing the semantic meaning of natural language is difficult for a machine. Meanwhile, many knowledge graph embedding methods have been proposed. The key idea is to represent each predicate/entity as a low-dimensional vector, such that the relation information in the KG could be preserved. The learned vectors could benefit various applications such as KG completion and recommender systems. In this paper, we explore to use them to handle the QA-KG problem. However, this remains a challenging task since a predicate could be expressed in different ways in natural language questions. Also, the ambiguity of entity names and partial names makes the number of possible answers large.

To bridge the gap, we propose an effective Knowledge Embedding based Question Answering (KEQA) framework. We focus on answering the most common types of questions, i.e., simple questions, in which each question could be answered by the machine straightforwardly if its single head entity and single predicate are correctly identified. To answer a simple question, instead of inferring its head entity and predicate directly, KEQA targets at jointly recovering the question's head entity, predicate, and tail entity representations in the KG embedding spaces. Based on a carefully-designed joint distance metric, the three learned vectors' closest fact in the KG is returned as the answer. Experiments on a widely-adopted benchmark demonstrate that the proposed KEQA outperforms the state-of-the-art QA-KG methods.

KEYWORDS

Question answering, knowledge graph embedding, deep learning

ACM Reference Format:
Xiao Huang, Jingyuan Zhang, Dingcheng Li, Ping Li. 2019. Knowledge Graph Embedding Based Question Answering. In *The Twelfth ACM International Conference on Web Search and Data Mining (WSDM '19), February 11–15, 2019, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3289600.3290956

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
WSDM '19, February 11–15, 2019, Melbourne, VIC, Australia
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-5940-5/19/02…$15.00
https://doi.org/10.1145/3289600.3290956

1 INTRODUCTION

With the rise of large-scale knowledge graphs such as Wikidata, Freebase [19], DBpedia [22], and YAGO [33], question answering (QA) over knowledge graph becomes a crucial topic and attracts massive attention [6, 27, 29]. A knowledge graph (KG) is a directed graph with real-world entities as nodes and their relations as edges [25, 36]. In this graph, each directed edge, along with its head entity and tail entity, constitute a triple, i.e., (*head entity*, *predicate*, *tail entity*), which is also named as a fact. Real-world knowledge graphs often contain millions or billions of facts[1]. Their large volume and complex data structures make it difficult for regular users to access the substantial and valuable knowledge in them. To bridge the gap, Question Answering over Knowledge Graph (QA-KG) is proposed [10, 21]. It targets at automatically translating the end users' natural language questions into structured queries such as SPARQL, and returning entities and/or predicates in the KG as answers. For example, given the question "Which Olympics was in Australia?", QA-KG aims to identify its corresponding two facts, i.e., (*Australia*, *olympics_participated_in*, *1952/2004 Summer Olympics*).

Question answering over knowledge graph provides a way for artificial intelligence systems to incorporate knowledge graphs as a key ingredient to answer human questions, with applications ranging from search engine design[2] to conversational agent building [20]. However, the QA-KG problem is far from solved since it involves multiple challenging subproblems such as semantic analysis [45] and entity linking [4, 30].

The effectiveness of knowledge graph embedding [7, 38] in different real-world applications [36] motivates us to explore its potential usage in solving the QA-KG problem. Knowledge graph embedding [26, 41] targets at learning a low-dimensional vector representation for each predicate/entity in a KG, such that the original relations are well preserved in the vectors. These learned vector representations could be employed to complete a variety of downstream applications efficiently. Examples include KG completion [25, 34], recommender systems [49], and relation extraction [20, 40]. In this paper, we propose to take advantage of the knowledge graph embedding to perform QA-KG. The KG embedding representations could advance the QA-KG in several ways. They not only are within a low-dimensional space, but also could promote the downstream applications to take the entire KG into consideration [49], because even a single predicate/entity representation is a result of interactions with the whole KG. In addition, similar predicates/entities tend to have similar vectors. This property could help the downstream algorithms handle predicates or entities that are not in the training data.

However, it remains a nontrivial task to conduct QA-KG based on the knowledge graph embedding [23]. There are three major

[1] https://www.wikidata.org/wiki/Wikidata:Statistics
[2] https://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html

challenges. First, a predicate often has various expressions in natural language questions [3, 45]. These expressions could be quite different from the predicate names. For instance, the predicate *person.nationality* can be expressed as "what is ... 's nationality", "which country is ... from", "where is ... from", etc. Second, even assuming that the entity names could be accurately identified, the ambiguity of entity names and partial names would still make it difficult to find the correct entity, since the number of candidates is often large. As the size of KG keeps increasing, many entities would share the same names. Also, end users could use partial names in their utterances. For example, in the question "How old is Obama?", only part of the entity name *Barack Obama* is indicated. Third, the domains of end users' questions are often unbounded, and any KG is far from complete [25]. New questions might involve predicates that are different from the ones in the training. This makes demands on the robustness of the QA-KG algorithms.

To bridge the gap, we explore how to take advantage of the knowledge graph embedding to perform question answering. In this paper, we focus on the most common type of questions [2, 13] in QA-KG, i.e., simple questions. A simple question is a natural language question that only involves a single head entity and a single predicate. Through analyzing the problem, we aim to answer three research questions. (i) How to apply the predicate embedding representations to bridge the gap between the natural language expressions and the KG's predicates? (ii) How to leverage the entity embedding representations to tackle the ambiguity challenge? (iii) How to take advantage of the global relations preserved in the KG embedding representations to advance the QA-KG framework? Following these questions, we propose a simple framework named Knowledge Embedding based Question Answering (KEQA). In summary, our key contributions are presented as follows.

- Formally define the knowledge graph embedding based question answering problem.
- Propose an effective framework KEQA that could answer a natural language question by jointly recovering its head entity, predicate, and tail entity representations in the knowledge graph embedding spaces.
- Design a joint distance metric that takes the structures and relations preserved in the knowledge graph embedding representations into consideration.
- Empirically demonstrate the effectiveness and robustness of KEQA on a large benchmark, i.e., SimpleQuestions.

2 PROBLEM STATEMENT

Notations: We use an uppercase bold letter to denote a matrix (e.g., $\mathbf{W}$) and a lowercase bold letter to represent a vector (e.g., $\mathbf{p}$). The i^{th} row of a matrix $\mathbf{P}$ is denoted as $\mathbf{p}_i$. The transpose of a vector is denoted as $\mathbf{p}^\top$. The ℓ^2 norm of a vector is denoted as $\|\mathbf{p}\|_2$. We use $\{\mathbf{p}_i\}$ to represent a sequence of vectors $\mathbf{p}_i$. The operation $\mathbf{s} = [\mathbf{x}; \mathbf{h}]$ denotes concatenating column vectors $\mathbf{x}$ and $\mathbf{h}$ into a new vector $\mathbf{s}$.

Definition 1 (Simple Question) [6] *If a natural language question only involves a single head entity and a single predicate in the knowledge graph, and takes their tail entity/entities as the answer, then this question is referred as a simple question.*

We summarize the important symbols in this paper in Table 1. We use (h, ℓ, t) to represent a fact, which means that there exists a relation ℓ from a head entity h to a tail entity t. Let $\mathcal{G}$ be a knowledge graph that consists of a large number of facts. The total numbers of predicates and entities are represented as M and N. The names of these predicates and entities are given. We apply a scalable KG embedding algorithm such as TransE [7] and TransR [25] to $\mathcal{G}$, and obtain the embedding representations of its predicates and entities denoted as $\mathbf{P}$ and $\mathbf{E}$ respectively. Thus, the vector representations of the i^{th} predicate and j^{th} entity are denoted as $\mathbf{p}_i$ and $\mathbf{e}_j$ respectively. The relation function defined by the KG embedding algorithm is $f(\cdot)$, i.e., given a fact (h, ℓ, t), we have $\mathbf{e}_t \approx f(\mathbf{e}_h, \mathbf{p}_\ell)$. Let $\mathcal{Q}$ be a set of simple questions. For each question in $\mathcal{Q}$, the corresponding head entity and predicate are given.

Table 1: The important symbols and their definitions.

Notations	Definitions
$\mathcal{G}$	a knowledge graph
(h, ℓ, t)	a fact, i.e., (head entity, predicate, tail entity)
$\mathcal{Q}$	a set of simple questions with ground truth facts
M	total number of predicates in $\mathcal{G}$
N	total number of entities in $\mathcal{G}$
d	dimension of the embedding representations
$\mathbf{P} \in \mathbb{R}^{M \times d}$	embedding representations of all predicates in $\mathcal{G}$
$\mathbf{E} \in \mathbb{R}^{N \times d}$	embedding representations of all entities in $\mathcal{G}$
$f(\cdot)$	relation function, given (h, ℓ, t), $\Rightarrow \mathbf{e}_t \approx f(\mathbf{e}_h, \mathbf{p}_\ell)$
$\hat{\mathbf{p}}_\ell \in \mathbb{R}^{1 \times d}$	predicted predicate representation
$\hat{\mathbf{e}}_h \in \mathbb{R}^{1 \times d}$	predicted head entity representation
HED	Head Entity Detection model
$\text{HED}_{\text{entity}}$	head entity name tokens returned by the HED
HED_{non}	non entity name tokens returned by the HED

The terminology *simple question* is defined in Definition 1. A simple question could be answered by the machine straightforwardly if its single head entity and single predicate are identified. Given the conditions described above, we now formally define the knowledge graph embedding based question answering problem.

> Given a knowledge graph $\mathcal{G}$ associated with all its predicates' and entities' names and embedding representations $\mathbf{P}$ & $\mathbf{E}$, the relation function $f(\cdot)$, as well as a set of simple questions $\mathcal{Q}$ associated with corresponding head entities and predicates, we aim to design an end-to-end framework that takes a new simple question as input and automatically returns the corresponding head entity and predicate. Performance of the framework is evaluated by the accuracy of predicting both head entity and predicate correctly.

3 KNOWLEDGE EMBEDDING BASED QA-KG

Simple questions constitute the majority of questions [2, 13] in the QA-KG problem. Each of them can be answered by the tail entity/entities if the correct head entity and predicate are identified. To accurately predict the head entity and predicate, we propose the Knowledge Embedding based Question Answering (KEQA) framework. Its main idea is illustrated in Figure 1. The KG $\mathcal{G}$ is already embedded into two low-dimensional spaces, and each fact (h, ℓ, t) could be represented as three latent vectors, i.e., $(\mathbf{e}_h, \mathbf{p}_\ell, \mathbf{e}_t)$. Thus, given a question, as long as we could predict its corresponding fact's $\mathbf{e}_h$ and $\mathbf{p}_\ell$, then this question could be answered correctly.

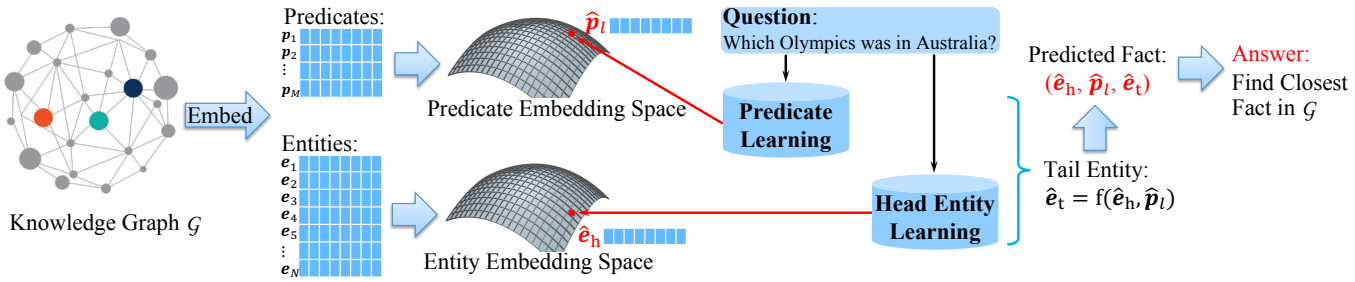

Figure 1: Instead of inferring the head entity and predicate directly, KEQA targets at jointly recovering the question's head entity, predicate, and tail entity representations $(\hat{\mathbf{e}}_h, \hat{\mathbf{p}}, \hat{\mathbf{e}}_t)$ in the knowledge graph embedding spaces.

KEQA achieves the goal via three steps. (i) Based on the questions in $\mathcal{Q}$ and their predicates' embedding representations, KEQA trains a predicate learning model that takes a question as the input and returns a vector $\hat{\mathbf{p}}_\ell$ that lies in the KG embedding space as the predicted predicate representation. Similarly, a head entity learning model could be constructed to predict the question's head entity representation $\hat{\mathbf{e}}_h$. (ii) Since the number of entities in a KG is often large, KEQA employs a Head Entity Detection model to reduce the candidate head entities. The main goal is to identify several tokens in the question as the predicted head entity name, then the search space is reduced from the entire entities to a number of entities with the same or similar names. Then $\hat{\mathbf{e}}_h$ is mainly used to tackle the ambiguity challenge. (iii) Given the relation function $f(\cdot)$ defined by the KG embedding algorithm, KEQA computes the predicted tail entity representation $\hat{\mathbf{e}}_t = f(\hat{\mathbf{e}}_h, \hat{\mathbf{p}}_\ell)$. Based on a carefully-designed joint distance metric, the predicted fact $(\hat{\mathbf{e}}_h, \hat{\mathbf{p}}_\ell, \hat{\mathbf{e}}_t)$'s closest fact in $\mathcal{G}$ is returned as the question's answer.

3.1 Knowledge Graph Embedding

The proposed framework KEQA employs the embedding representations of all predicates $\mathbf{P}$ and entities $\mathbf{E}$ as the infrastructure. We utilize an existing KG embedding algorithm to learn $\mathbf{P}$ and $\mathbf{E}$.

Knowledge graph embedding [8, 36] aims to represent each predicate/entity in a KG as a low-dimensional vector, such that the original structures and relations in the KG are preserved in these learned vectors. The core idea of most of the existing KG embedding methods [7, 24, 25, 38–41] could be summarized as follows. For each fact (h, ℓ, t) in $\mathcal{G}$, we denote its embedding representations as $(\mathbf{e}_h, \mathbf{p}_\ell, \mathbf{e}_t)$. The embedding algorithm initializes the values of $\mathbf{e}_h$, $\mathbf{p}_\ell$, and $\mathbf{e}_t$ randomly [7, 14] or based on the trained word embedding models [26, 32]. Then, a function $f(\cdot)$ that measures the relation of a fact (h, ℓ, t) in the embedding spaces is defined, i.e., $\mathbf{e}_t \approx f(\mathbf{e}_h, \mathbf{p}_\ell)$. For example, TransE [7] defines the relation as $\mathbf{e}_t \approx \mathbf{e}_h + \mathbf{p}_\ell$ and TransR [25] defines it as $\mathbf{e}_t\mathbf{M}_\ell \approx \mathbf{e}_h\mathbf{M}_\ell + \mathbf{p}_\ell$, where $\mathbf{M}_\ell$ is a transform matrix of predicate ℓ. Finally, the embedding algorithm minimizes the overall distance between $\mathbf{e}_t$ and $f(\mathbf{e}_h, \mathbf{p}_\ell)$, for all the facts in $\mathcal{G}$. A typical way is to define a margin-based ranking criterion and train on both positive and negative samples, i.e., facts and synthetic facts that do not exist in $\mathcal{G}$.

As shown in Figure 1, we define the surface where the learned predicate representations $\{\mathbf{p}_i\}$ for $i = 1, \ldots, M$ lie in, as the predicate embedding space. The surface where $\{\mathbf{e}_i\}$ for $i = 1, \ldots, N$ lie in is denoted as the entity embedding space.

3.2 Predicate and Head Entity Learning Models

Given a simple question, our goal is to find a point in the predicate embedding space as its predicate representation $\hat{\mathbf{p}}_\ell$, and a point in the entity embedding space as its head entity representations $\hat{\mathbf{e}}_h$.

For all the questions that can be answered by $\mathcal{G}$, their predicates' vector representations must lie in the predicate embedding space. Thus, we aim to design a model that takes a question as the input and returns a vector $\hat{\mathbf{p}}_\ell$ that is as close as possible to this question's predicate embedding representation $\mathbf{p}_\ell$. To achieve this goal, a simple neural network architecture is employed, as shown in Figure 2. It mainly consists of a bidirectional recurrent neural network layer and an attention layer. The core idea is to take the order and the importance of words into consideration. Words with different orders could have different meanings, and the importance of words could be different. For example, the entity name related words in a question often have less contribution to the predicate learning model.

3.2.1 Neural Network Based Predicate Representation Learning. To predict the predicate of a question, a traditional solution is to learn the mapping based on the semantic parsing and manually-created

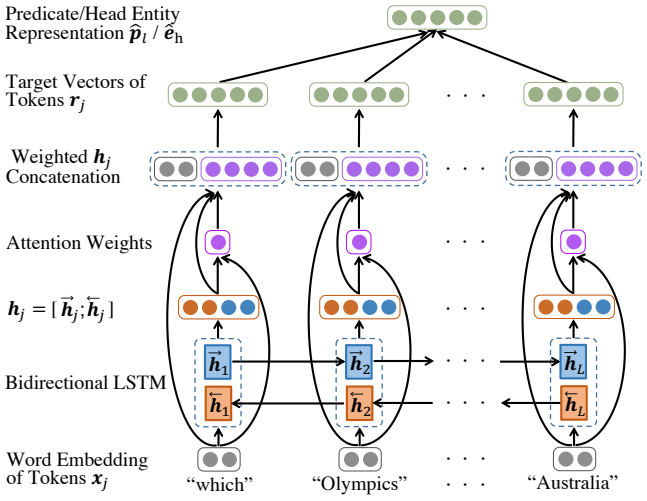

Figure 2: The architecture of the proposed predicate and head entity learning models.

lexicons [3], or simply consider each type of predicate as a label category to transform it into a classification problem [29, 35]. However, since the domains of end users' questions are often unbounded, a new question's predicate might be different from all the ones in the training data Q. The traditional solutions could not handle this scenario. In addition, we observe that the global relation information preserved in $\mathbf{P}$ and $\mathbf{E}$ is available and could be potentially used to improve the overall question answering accuracy. To bridge the gap, we develop a predicate learning model based on neural networks.

With the long short-term memory (LSTM) [1] as a typical example of the recurrent neural network, Figure 2 illustrates the architecture of our proposed solution. Given a question with length L, we first map its L tokens into a sequence of word embedding vectors $\{\mathbf{x}_j\}$, for $j = 1, \ldots, L$, based on a pre-trained model such as GloVe [31]. Then we employ a bidirectional LSTM [1] to learn a forward hidden state sequence $(\overrightarrow{\mathbf{h}}_1, \overrightarrow{\mathbf{h}}_2, \ldots, \overrightarrow{\mathbf{h}}_L)$ and a backward hidden state sequence $(\overleftarrow{\mathbf{h}}_1, \overleftarrow{\mathbf{h}}_2, \ldots, \overleftarrow{\mathbf{h}}_L)$. Taking the backward one as an example, $\{\overleftarrow{\mathbf{h}}_j\}$ are computed via the following equations.

$$\mathbf{f}_j = \sigma(\mathbf{W}_{xf}\mathbf{x}_j + \mathbf{W}_{hf}\overleftarrow{\mathbf{h}}_{j+1} + \mathbf{b}_f), \tag{1}$$

$$\mathbf{i}_j = \sigma(\mathbf{W}_{xi}\mathbf{x}_j + \mathbf{W}_{hi}\overleftarrow{\mathbf{h}}_{j+1} + \mathbf{b}_i), \tag{2}$$

$$\mathbf{o}_j = \sigma(\mathbf{W}_{xo}\mathbf{x}_j + \mathbf{W}_{ho}\overleftarrow{\mathbf{h}}_{j+1} + \mathbf{b}_o), \tag{3}$$

$$\mathbf{c}_j = \mathbf{f}_j \circ \mathbf{c}_{j+1} + \mathbf{i}_j \tanh(\mathbf{W}_{xc}\mathbf{x}_j + \mathbf{W}_{hc}\overleftarrow{\mathbf{h}}_{j+1} + \mathbf{b}_c), \tag{4}$$

$$\overleftarrow{\mathbf{h}}_j = \mathbf{o}_j \circ \tanh(\mathbf{c}_j), \tag{5}$$

where $\mathbf{f}_j$, $\mathbf{i}_j$, and $\mathbf{o}_j$ are the forget, input, and output gates' activation vectors respectively. $\mathbf{c}_j$ is the cell state vector. σ and tanh are the sigmoid and Hyperbolic tangent functions. $\circ$ denotes the Hadamard product. We concatenate the forward and backward hidden state vectors and obtain $\mathbf{h}_j = [\overrightarrow{\mathbf{h}}_j; \overleftarrow{\mathbf{h}}_j]$.

The attention weight of the j^{th} token, i.e., α_j, is calculated based on the following formulas.

$$\alpha_j = \frac{\exp(q_j)}{\sum_{i=1}^{L}\exp(q_i)}, \tag{6}$$

$$q_j = \tanh(\mathbf{w}^\top[\mathbf{x}_j; \mathbf{h}_j] + b_q). \tag{7}$$

We apply the attention weight α_j to $\mathbf{h}_j$ and concatenate it with the word embedding $\mathbf{x}_j$, resulting a hidden state $\mathbf{s}_j = [\mathbf{x}_j; \alpha_j\mathbf{h}_j]$. A fully connected layer is then applied to $\mathbf{s}_j$, and its result $\mathbf{r}_j \in \mathbb{R}^{d\times 1}$ is denoted as the target vector of the j^{th} token. The predicted predicate representation $\hat{\mathbf{p}}_\ell$ is computed as the mean of all tokens' target vectors, that is,

$$\hat{\mathbf{p}}_\ell = \frac{1}{L}\sum_{j=1}^{L}\mathbf{r}_j^\top. \tag{8}$$

All the weight matrices, weight vector $\mathbf{w}$, and bias terms are calculated based on the training data, i.e., questions in Q and their predicates' embedding representations.

3.2.2 Neural Network based Head Entity Learning Model. Given a question, instead of inferring the head entity directly, we target at recovering its representation in the KG embedding space. Thus, the goal of the head entity learning model is to compute a vector $\hat{\mathbf{e}}_h$ that is as close as possible to this question's head entity embedding representation. Similar to the computation of $\hat{\mathbf{p}}_\ell$, we use the same neural network architecture in Figure 2 to obtain the predicted head entity representation $\hat{\mathbf{e}}_h$.

However, the number of entities in a KG is often large, and it could be expensive and noisy when comparing $\hat{\mathbf{e}}_h$ with all entity embedding representations in $\mathbf{E}$. To make the learning more efficient and effective, KEQA employs a head entity detection model to reduce the number of candidate head entities.

3.3 Head Entity Detection Model

In this step, our goal is to select one or several successive tokens in a question, as the name of the head entity, such that the search space could be reduced from the entire entities to a number of entities with the same or similar names. Then the main role of $\hat{\mathbf{e}}_h$ would become handling the ambiguity challenge.

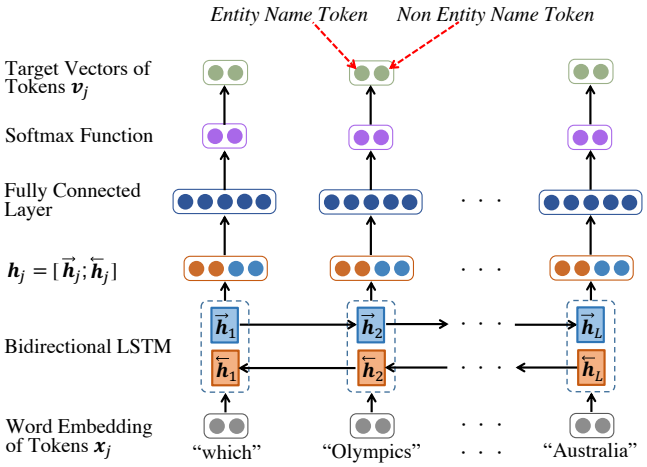

Figure 3: Structure of Head Entity Detection (HED) model.

To make our framework simple, we employ a bidirectional recurrent neural network (e.g., LSTM) based model to perform the head entity token detection task. The architecture of this Head Entity Detection (HED) model is shown in Figure 3. It has a similar structure to the one in predicate/head entity learning model, but without the attention layer. We first map the question into a sequence of word embedding vectors $\{\mathbf{x}_j\}$, for $j = 1, \ldots, L$, and then apply a bidirectional recurrent neural network to $\mathbf{x}_j$ to learn $\mathbf{h}_j = [\overrightarrow{\mathbf{h}}_j; \overleftarrow{\mathbf{h}}_j]$. A fully connected layer and a softmax function are then applied to $\mathbf{h}_j$, resulting the target vector $\mathbf{v}_j \in \mathbb{R}^{2\times 1}$. The two values in $\mathbf{v}_j$ are corresponding to the probabilities that the j^{th} token belongs to the two label categories, i.e., *entity name token* and *non entity name token*. In such a way, we classify each token and recognize one or several tokens as the head entity name. We denote these tokens as $\text{HED}_{\text{entity}}$, and the remaining tokens in the question as HED_{non}.

We use the questions in Q and their head entity names as the training data to train the HED model. Since entity name tokens in these questions are successive, the trained model would also return successive tokens as $\text{HED}_{\text{entity}}$ with a high probability. If discrete $\text{HED}_{\text{entity}}$ is returned, then each successive part would be considered as an independent head entity name. It should be noted

that $\text{HED}_{\text{entity}}$ might be only part of the correct head entity name. Thus, all entities that are the same as or contain $\text{HED}_{\text{entity}}$ would be included as the candidate head entities, which might still be large since many entities would share the same names in a large KG.

3.4 Joint Search on Embedding Spaces

For each new simple question, we have predicted its predicate and head entity representations $\hat{\mathbf{p}}_\ell$ and $\hat{\mathbf{e}}_h$, as well as its candidate head entities. Our goal is to find a fact in $\mathcal{G}$ that matches these learned representations and candidates the most.

3.4.1 Joint Distance Metric. If a fact's head entity belongs to the candidate head entities, we name it as a candidate fact. Let C be a set that collects all the candidate facts. To measure the distance between a candidate fact (h, ℓ, t) and the predicted representations $(\hat{\mathbf{e}}_h, \hat{\mathbf{p}}_\ell)$, an intuitive solution is to represent (h, ℓ, t) as $(\mathbf{e}_h, \mathbf{p}_\ell)$ and define the distance metric as the sum of the distance between $\mathbf{e}_h$ and $\hat{\mathbf{e}}_h$ and distance between $\mathbf{p}_\ell$ and $\hat{\mathbf{p}}_\ell$. This solution, however, does not take the meaningful relation information preserved in the KG embedding representations into consideration.

We propose a joint distance metric by taking advantage of the relation information $\mathbf{e}_t \approx f(\mathbf{e}_h, \mathbf{p}_\ell)$. Mathematically, the proposed joint distance metric is defined as,

$$\underset{(h,\ell,t)\in C}{\text{minimize}} \quad \|\mathbf{p}_\ell - \hat{\mathbf{p}}_\ell\|_2 + \beta_1\|\mathbf{e}_h - \hat{\mathbf{e}}_h\|_2 + \beta_2\|f(\mathbf{e}_h, \mathbf{p}_\ell) - \hat{\mathbf{e}}_t\|_2 - \beta_3 sim[n(h), \text{HED}_{\text{entity}}] - \beta_4 sim[n(\ell), \text{HED}_{\text{non}}], \quad (9)$$

where $\hat{\mathbf{e}}_t = f(\hat{\mathbf{e}}_h, \hat{\mathbf{p}}_\ell)$. Function $n(\cdot)$ returns the name of the entity or predicate. $\text{HED}_{\text{entity}}$ and HED_{non} denote the tokens that are classified as entity name and non entity name by the HED model. Function $sim[\cdot, \cdot]$ measures the similarity of two strings. β_1, β_2, β_3, and β_4 are predefined weights to balance the contribution of each term. In this paper, we use ℓ^2 norm to measure the distance, and it is straightforward to extend to other vector distance measures.

The first three terms in Eq. (9) measure the distance between a fact (h, ℓ, t) and our prediction in the KG embedding spaces. We use $f(\mathbf{e}_h, \mathbf{p}_\ell)$ to represent the tail entity's embedding vector, instead of $\mathbf{e}_t$. It is because in a KG, there might be several facts that have the same head entity and predicate, but different tail entities. Thus, a single tail entity $\mathbf{e}_t$ might not be able to answer the question. Meanwhile, $f(\mathbf{e}_h, \mathbf{p}_\ell)$ matches the predicted tail entity $\hat{\mathbf{e}}_t$ since it is also inferred based on $f(\cdot)$. We tend to select a fact with head entity name exactly the same as $\text{HED}_{\text{entity}}$, and with predicate name mentioned by the question. We achieve these two goals via the fourth and fifth terms in Eq. (9) respectively. The fact (h^*, ℓ^*, t^*) that minimizes the objective function is returned.

3.4.2 Knowledge Embedding based Question Answering. The entire processes of KEQA is summarized in Algorithm 1. Given a KG $\mathcal{G}$ and a question set $\mathcal{Q}$ with corresponding answers, we train a predicate learning model, a head entity learning model, and a HED model, as shown from line 1 to line 9. Then, for any new simple question Q, we input it into the trained predicate learning model, head entity learning model, and HED model to learn its predicted predicate representation $\hat{\mathbf{p}}_\ell$, head entity representation $\hat{\mathbf{e}}_h$, entity name tokens $\text{HED}_{\text{entity}}$, and non entity name tokens HED_{non}. Based on the learned entity name/names in $\text{HED}_{\text{entity}}$, we search the entire $\mathcal{G}$ to find the candidate fact set C. For all facts in C, we compute their joint distance to the predicted representations $(\hat{\mathbf{e}}_h, \hat{\mathbf{p}}_\ell, \hat{\mathbf{e}}_t)$ based on the objective function in Eq. (9). The fact (h^*, ℓ^*, t^*) with the minimum distance is selected. Finally, we return the head entity h^* and predicate ℓ^* as the answer of Q.

Algorithm 1: The proposed KEQA framework

Input: $\mathcal{G}$, predicates' and entities' names, $\mathbf{P}$, $\mathbf{E}$, $\mathcal{Q}$, a new simple question Q.
Output: head entity h^* and predicate ℓ^*.

```
/* Training the predicate learning model:        */
```
1 **for** Q_i *in* $\mathcal{Q}$ **do**
2 Take the L tokens of Q_i as the input and its predicate ℓ as the label to train, as shown in Figure 2;
3 Update weight matrices $\{\mathbf{W}\}$, $\mathbf{w}$, $\{\mathbf{b}\}$, and b_q to minimize the objective function $\|\mathbf{p}_\ell - \frac{1}{L}\sum_{j=1}^{L}\mathbf{r}_j^\top\|_2$;

```
/* Training the head entity learning model:      */
```
4 **for** Q_i *in* $\mathcal{Q}$ **do**
5 Take the L tokens of Q_i as the input and its head entity h as the label to train, as shown in Figure 2;
6 Update weight matrices and bias terms to minimize the objective function $\|\mathbf{e}_h - \frac{1}{L}\sum_{j=1}^{L}\mathbf{r}_j^\top\|_2$;

```
/* Training the HED model:                       */
```
7 **for** Q_i *in* $\mathcal{Q}$ **do**
8 Take the L tokens of Q_i as the input and its head entity name positions as the label to train;
9 Update weight matrices and bias as shown in Figure 3;

```
/* Question answering processes:                 */
```
10 Input Q into the predicate learning model to learn $\hat{\mathbf{p}}_\ell$;
11 Input Q into the head entity learning model to learn $\hat{\mathbf{e}}_h$;
12 Input Q into the HED model to learn $\text{HED}_{\text{entity}}$ and HED_{non};
13 Find the candidate fact set C from $\mathcal{G}$, based on $\text{HED}_{\text{entity}}$;
14 For all facts in C, calculate the fact (h^*, ℓ^*, t^*) that minimizes the objective function in Eq. (9).

In summary, the proposed framework KEQA enjoys several nice properties. First, by performing question answering based on the KG embedding, KEQA is able to handle questions with predicates and entities that are different from all the ones in the training data. Second, by taking advantage of the structure and relation information preserved in the KG embedding representations, KEQA could perform the head entity, predicate, and tail entity predictions jointly. The three subtasks would mutually complement each other. Third, KEQA is generalizable to different KG embedding algorithms. Thus, the performance of KEQA might be further improved by more sophisticated KG embedding algorithms.

4 EXPERIMENTS

We evaluate the effectiveness and generalizability of the proposed framework KEQA on a large QA-KG benchmark. In this section, we aim to study the following three research questions:

- **Q1**. How effective is KEQA compared with the state-of-the-art QA-KG methods w.r.t. different freebase subsets?

- **Q2**. How does the performance of KEQA vary when different KG embedding algorithms are employed?
- **Q3**. The objective function of KEQA consists of five terms as shown in Eq. (9). How much does each term contribute?

4.1 Datasets

We first introduce the knowledge graph subsets and question answering dataset used in the experiments. All the data are publicly available. Their statistics are shown in Table 2.

Table 2: The statistics of the question answering datasets.

	FB2M	FB5M	SimpleQuestions
# Training	14,174,246	17,872,174	75,910
# Validation	N.A.	N.A.	10,845
# Test	N.A.	N.A.	21,687
# Predicates (M)	6,701	7,523	1,837
# Entities (N)	1,963,130	3,988,105	131,681
Vocabulary Size	733,278	1,213,205	61,336

FB2M and **FB5M** [19]: Freebase is often regarded as a reliable KG since it is collected and trimmed mainly by the community members. Two large subsets of freebase are employed in this paper, i.e., FB2M and FB5M. Their predicate number M and entity number N are list in Table 2. The repeated facts have been deleted. The application programming interface (API) of freebase is no long available. Thus, we use an entity name collection[3] to build the mapping between entities and their names.

SimpleQuestions [6]: It contains more than ten thousand simple questions associated with corresponding facts. All these facts belong to FB2M. All questions are phrased by English speakers based on the facts and their context. It has been used as the benchmark for the recent QA-KG methods [6, 18, 29].

4.2 Experimental Settings

To evaluate the performance of the QA-KG methods, we follow the traditional settings [10, 27, 46] and use the same training, validation, and test splits that are originally provided in SimpleQuestions [6]. Either FB2M or FB5M is employed as the KG $\mathcal{G}$. Then a KG embedding algorithm such as TransE [7] and TransR [25] is applied to $\mathcal{G}$ to learn the **P** and **E**. It should be noted that **P** and **E** are not extra information sources. Then, a QA-KG method is applied to predict the head entity and predicate of each question in the test split. Its performance is measured by the accuracy of predicting both head entity and predicate correctly.

As claimed in our formal problem definition, the evaluation criterion is defined as the accuracy of predicting a new question' both head entity and predicate correctly. The dimension of the KG embedding representations d is set to be 250. A pre-trained word embedding based on GloVe [31] is used. To measure the similarity of two string, i.e., to build the function $sim[\cdot,\cdot]$, we use the implementation Fuzzy[4]. If it is not specific, the KG embedding algorithm TransE [7] would be employed to learn the embedding representations of all predicates **P** and entities **E**.

[3]https://github.com/zihangdai/CFO
[4]https://pypi.org/project/Fuzzy/

4.3 Effectiveness of KEQA

We now answer the first research question asked at the beginning of this section, i.e., how effective is KEQA. We include 7 state-of-the-art QA-KG algorithms and one variation of KEQA as the baselines:

- *Bordes et al.* [6]: It learns latent representations for words, predicates, and entities, based on the training questions, such that a new question and candidate facts could be projected into the same space and compared.
- *Dai et al.* [10]: It employs a bidirectional gated recurrent units based neural network to rank the candidate predicates. Suggestions from the freebase API are used.
- *Yin et al.* [46]: It employs a character-level convolutional neural network to match the questions and predicates.
- *Golub and He* [18]: It designs a character-level and attention-based LSTM to encode and decode questions.
- *Bao et al.* [2]: It manually defines several types of constraints and performs constraint learning to handle complex questions, in which each question is related to several facts. Extra training questions and freebase API are used.
- *Lukovnikov et al.* [27]: It utilizes a character-level gated recurrent units neural network to project questions and predicates/entities into the same space.
- *Mohammed et al.* [29]: It treats the predicate prediction as a classification problem and uses different neural networks to solve it. It performs entity linking based on Fuzzy[4].
- *KEQA_noEmbed*: No KG embedding algorithm is used. Instead, it generates the predicate and entity embedding representations **P** and **E** randomly.

As shown in the introduction above, all the baselines have taken advantage of deep learning models to advance their methods. We use their results reported in the corresponding papers or the authors' implementations. The performance of different methods on SimpleQuestions w.r.t. FB2M and FB5M is listed in Table 3.

Table 3: Performance of all methods on SimpleQuestions.

	FB2M (Accuracy)	FB5M
Bordes et al. (2015) [6]	0.627	0.639
Dai et al.[3] (2016) [10]	N.A.	0.626
Yin et al. (2016) [46]	0.683 (+8.9%)	0.672
Golub and He (2016) [18]	0.709 (+13.1%)	0.703
Bao et al. (2016) [2]	0.728 (+16.1%)	Entire Freebase
Lukovnikov et al. (2017) [27]	0.712 (+13.6%)	N.A.
Mohammed et al.[5](2018) [29]	0.732 (+16.7%)	N.A.
KEQA_noEmbed	0.731 (+16.6%)	0.726
KEQA	**0.754** (+**20.3**%)	**0.749**

As mentioned by several other work [27, 29], a few algorithms [10, 46] achieve high accuracy, but they either used extra information sources or have no available implementations [35, 47]. The extra training data freebase API suggestions, freebase entity linking results, and trained segmentation models. These rely on the freebase API, which is no longer available. Instead, our framework KEQA uses an entity name collection[3], which is incomplete. Thus, for Dai

[5]https://github.com/castorini/BuboQA/tree/master/evidence_integration

et al. [10] and Yin et al. [46], we report their results when no extra training data is used. There are two work [35, 47] claimed much higher accuracy, but without publicly available implementations. We are not able to replicate them, which has also been pointed out by other work [29].

From the results in Table 3, we have three observations. First, the proposed framework KEQA outperforms all the baselines. KEQA achieves 20.3% improvement comparing to the accuracy when SimpleQuestions was released [6]. Second, KEQA achieves 3.1% higher accuracy compared to KEQA_noEmbed. It demonstrates that the separate task KG embedding indeed could help the question answering task. Third, the performance of KEQA decreases 0.7% when applied to FB5M. It is because all the ground truth facts belong to FB2M [6], and FB5M has 26.1% more facts than FB2M.

By jointly predicting the question's predicate and head entity, KEQA achieves an accuracy of 0.754. In the predicate prediction subtask, KEQA achieves an accuracy of 0.815 on the validation split, which is worse than the most recent one 0.828 achieved by Mohammed et al. [29]. This gap suggests that our framework might be further improved by a more sophisticated model. Nevertheless, KEQA still outperforms Mohammed et al. [29] in the simple question answering task. This confirms the effectiveness of our proposed jointly learning framework. Through the jointly learning, KEQA achieves an accuracy of 0.816 in predicting the head entity, 0.754 in predicting both head entity and predicate, and 0.680 in predicting the entire fact, on the test split and FB2M. It implies that some of the ground truth facts do not exist in FB2M.

4.4 Generalizability and Robustness Evaluation

4.4.1 Generalizability of KEQA. To study how general is KEQA when different KG embedding algorithms are used, we include three scalable KG embedding methods in the comparison. Detailed introductions are listed as follows.

- *KEQA_TransE*: TransE [7] is used to perform the KG embedding. It is a typical translation-based method. It defines the relation function as $\mathbf{e}_t \approx f(\mathbf{e}_h, \mathbf{p}_\ell) = \mathbf{e}_h + \mathbf{p}_\ell$, and then performs the margin-based ranking to make all the facts approach to satisfy the relation function.
- *KEQA_TransH*: TransH [39] is used to perform the KG embedding. TransH is similar to TransE, and defines the relation function as $\mathbf{e}_t^\perp \approx \mathbf{e}_h^\perp + \mathbf{p}_\ell$, where $\mathbf{e}_t^\perp = \mathbf{e}_t - \mathbf{m}_\ell^\top \mathbf{e}_t \mathbf{m}_\ell$ and $\mathbf{m}_\ell$ is the hyperplane of predicate ℓ.
- *KEQA_TransR*: TransR [25] is similar to TransE, and defines the relation function as $\mathbf{e}_t \mathbf{M}_\ell \approx \mathbf{e}_h \mathbf{M}_\ell + \mathbf{p}_\ell$, where $\mathbf{M}_\ell$ is a transform matrix of ℓ.

The performance of KEQA when not using the KG embedding and when using different KG embedding algorithms is shown in Table 4. From the results, we have three major observations. First, the KG embedding algorithms have improved the performance of KEQA. For example, KEQA achieves 3.1% improvement when it is based on TransE, comparing to *KEQA_noEmbed*. Second, KEQA has similar performance when using different KG embedding algorithms. It demonstrates the generalizability of KEQA. Third, even when not using the KG embedding, KEQA could still achieve comparable performance to the state-of-the-art QA-KG methods as shown in Table 3. It validates the robustness of KEQA. The reason that randomly-generated $\mathbf{P}$ and $\mathbf{E}$ could achieve comparable performance is that it tends to make all $\{\mathbf{p}_\ell\}$ uniformly distributed and far away from each other. This would convert the representation prediction problem to a one that is similar to the classification task.

4.4.2 Robustness of KEQA. To further validate the robustness of KEQA, we reshuffle all the 108,442 questions in SimpleQuestions and get a new dataset named SimpleQ_Missing. To perform the reshuffle, we randomly split all the types of predicates into three groups, and assign questions to these groups based on the predicates. Thus, in SimpleQ_Missing, all the corresponding predicates of the questions in the test split have never been mentioned in the training and validation splits. In the end, we get 75,474 questions in the training split, 11,017 questions in the validation split, and 21,951 questions in the test split, which are roughly the same ratios as the ones in SimpleQuestions. The performance of KEQA with different KG embedding algorithms on SimpleQ_Missing is shown in Table 4.

Table 4: The performance of KEQA with different knowledge graph embedding algorithm on FB2M.

	SimpleQuestions	SimpleQ_Missing
KEQA_noEmbed	0.731	0.386
KEQA_TransE	0.754 (+3.1%)	0.418 (+8.3%)
KEQA_TransH	0.749 (+2.5%)	0.411 (+6.5%)
KEQA_TransR	0.753 (+3.0%)	0.417 (+8.0%)

From the results in Table 4, we observe that KEQA could still achieve an accuracy of 0.418 with the help of TransE. The global relation and structure information preserved in the KG embedding representations $\mathbf{P}$ and $\mathbf{E}$ enables KEQA to perform 8.3% better than Random. These observations demonstrate the robustness of KEQA.

4.5 Parameter Analysis

We now investigate how much could each term in the objective function of KEQA contribute. There are five terms in our objective function as shown in Eq. (9). We valid the performance of KEQA w.r.t. three groups of different combinations of terms. To study the contribution of every single term in Eq. (9), in the first group, i.e., *Only_Keep*, we only keep one of the five terms as the new objective function. To study the impact of missing one of the five terms, in the second group, i.e., *Remove*, we remove one of the five terms. To study the accumulated contributions, in the third group, i.e., *Accumulate*, we add terms as the new objective function one by one. The performance of KEQA w.r.t. different groups of objective functions on FB2M is summarized in Table 5.

Table 5: The performance of KEQA with different objective functions on FB2M.

	Only_Keep	Remove	Accumulate
$\|\mathbf{p}_\ell - \hat{\mathbf{p}}_\ell\|_2$	0.728	0.701	0.728
$\|\mathbf{e}_h - \hat{\mathbf{e}}_h\|_2$	0.195	0.751	0.745
$\|f(\mathbf{e}_h, \mathbf{p}_\ell) - \hat{\mathbf{e}}_t\|_2$	0.730	0.753	0.745
$sim[n(h), \mathrm{HED}_{\mathrm{entity}}]$	0.173	0.754	0.746
$sim[n(\ell), \mathrm{HED}_{\mathrm{non}}]$	0.435	0.746	0.754

From the results in Table 5, we have three major observations. First, the predicted predicate representation $\hat{\mathbf{p}}_\ell$ has the most significant contribution in our framework. The first term achieves an accuracy of 0.728 independently. It is because the number of predicates 1,837 is much smaller than the number of training questions 75,910. Second, the predicted head entity representation $\hat{\mathbf{e}}_h$ could complement $\hat{\mathbf{p}}_\ell$ in the joint learning. The accuracy increases from 0.728 to 0.745 when $\hat{\mathbf{e}}_h$ is used. The second term achieves a low accuracy independently since the total number of entities N is too large, e.g., N = 1,963,115 in FB2M. Third, the predicate name $n(\ell)$ improves the performance of the KEQA by 1.1%. It could be explained by the fact that some utterances share a few words with the corresponding predicate names.

5 RELATED WORK

Embedding-based question answering over KG attracts lots of attention recently. It is related to but different from our proposed KG embedding based question answering problem. The former relies on low-dimensional representations that are learned during the training of the QA-KG methods. The latter performs KG embedding to learn the low-dimensional representations first, and then conducts the QA-KG task. Yih et al. [45] and Bao et al. [2] reformulated the question answering problem as the generation of particular subgraphs. A series of work [5, 6, 9, 11, 12, 21, 27, 43, 44] proposed to project questions and candidate answers (or entire facts) into a unified low-dimensional space based on the training questions, and measure their matching scores by the similarities between their low-dimensional representations. Bordes et al. [5, 6, 9] achieved this projection by learning low-dimensional representations for all words, predicates, and entities, based on the training questions and paraphrases [16] of questions. Yang et al. [43, 44] achieved this projection by using the logical properties of questions and potential facts, such as semantic embedding and entity types. Several deep learning based models [10, 12, 21, 27, 46] achieved this projection by feeding words in questions into convolutional neural networks [12, 46], LSTM networks [18, 21], or gated recurrent units neural networks [10, 27]. Das et al. [11] achieved this projection by using matrix factorization to incorporate the corpus into the KG, and LSTM to embed a question. Most of these models rely on the margin-based ranking objective functions to learn the model weights. There are also several work [15, 18, 27, 46] explored to leverage the character-level neural networks to advance the performance. Most recently, Mohammed et al. [29] and Ture et al. [35] considered each predicate as a label category, and performed predicate linking via deep classification models.

Knowledge graph embedding targets at representing the high-dimensional KG as latent predicate and entity representations $\mathbf{P}$ and $\mathbf{E}$. Bordes et al. [8] achieved this goal by constructing two transform matrices $\mathbf{M}_{\text{head}}$ and $\mathbf{M}_{\text{tail}}$ for each type of predicate ℓ, and minimizing the distance between projections $\mathbf{M}_{\text{head}}\mathbf{e}_h$ and $\mathbf{M}_{\text{tail}}\mathbf{e}_t$ for all facts (h, ℓ, t) with ℓ as predicate. Bordes et al. [7] designed a translation-based model TransE. It trains two matrices $\mathbf{P}$ and $\mathbf{E}$, aiming to minimize the overall distance $\sum \|\mathbf{e}_h + \mathbf{p}_\ell - \mathbf{e}_t\|_2^2$ for all facts (h, ℓ, t). Motivated by TransE, a series of translation-based models [24, 25, 39] have been explored. Wang et al. [39] proposed TransH to handle one-to-many or many-to-one relations. Instead of measuring the distance between $\mathbf{e}_h$ and $\mathbf{e}_t$ directly, TransH projects them into a predicate-specific hyperplane. Lin et al. [25] proposed TransR, which defines a transform matrix $\mathbf{M}_\ell$ for each predicate ℓ and targets at minimizing $\sum \|\mathbf{e}_h\mathbf{M}_\ell + \mathbf{p}_\ell - \mathbf{e}_t\mathbf{M}_\ell\|_2^2$. Lin et al. [24] proposed PTransE, which advances TransE via taking multi-hop relations into consideration.

Efforts have also been devoted to incorporating the semantic information in a corpus into KG embedding. Socher et al. [32] and Long et al. [26] demonstrated that using pre-trained word embedding to initialize KG embedding methods would enhance the performance. Several work [14, 40, 41] explored to advance TransE, either via taking relation mentions in corpus into consideration [14, 40], or via projecting predicate/entity representations into a semantic hyperplane learned from the topic model [41]. Attempts [37, 38, 50] have also been made to apply TransE and word2vec [28] to model a KG and a corpus respectively, and then fuse them based on anchors in Wikipedia [38], entity descriptions [50], or contextual words of predicates/entities learned from the corpus [37]. Zhang et al. [48] jointly embedded the KG and corpus via negative sampling [28]. Xie et al. [42] and Fan et al. [17] explored the semantic information in entity descriptions to advance KG embedding.

6 CONCLUSIONS AND FUTURE WORK

Question answering over knowledge graph is a crucial problem since it enables regular users to easily access the valuable but complex information in the large knowledge graphs via natural language. It is also a challenging problem since a predicate could have different natural language expressions. It is hard for a machine to capture their semantic information. In addition, even assuming that the entity name of a question is correctly identified, the ambiguity of entity names and partial names would still make the number of candidate entities large.

To bridge the gap, we investigate a novel knowledge graph embedding based question answering problem and design a simple and effective framework KEQA. It targets at solving simple questions, i.e., the most common type of question in QA-KG. Instead of inferring the head entity and predicate directly, KEQA proposes to jointly recover the question's head entity, predicate, and tail entity representations in the KG embedding spaces. Attention-based bidirectional LSTM models are employed to perform the predicate and head entity representation learning. Since it is expensive and noisy to comparing with all entities in a KG, a head entity detection model is used to select successive tokens in a question as the name of the head entity, such that candidate head entity set would be reduced to a number of entities with the same or similar names. Given the predicted fact $(\hat{\mathbf{e}}_h, \hat{\mathbf{p}}_\ell, \hat{\mathbf{e}}_t)$, a carefully-designed joint distance metric is used to measure its distances to all candidate facts. The fact with the minimum distance is returned as the answer. Experiments on a large benchmark demonstrate that KEQA achieves better performance than all state-of-the-art methods.

In future work, we plan to study the follow-up open problems. (i) KEQA performs the question answering based on the pre-trained KG embedding. How can we advance it by jointly conducting the KG embedding and question answering? (ii) Real-world knowledge graphs and training questions are often updated dynamically. How can we extend our framework to handle this scenario?

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*.

[2] Junwei Bao, Nan Duan, Zhao Yan, Ming Zhou, and Tiejun Zhao. 2016. Constraint-Based Question Answering with Knowledge Graph. In *COLING*. 2503–2514.

[3] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic Parsing on Freebase from Question-Answer Pairs. In *EMNLP*. 1533–1544.

[4] Roi Blanco, Giuseppe Ottaviano, and Edgar Meij. 2015. Fast and Space-Efficient Entity Linking for Queries. In *WSDM*. 179–188.

[5] Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question Answering with Subgraph Embeddings. In *EMNLP*. 615–620.

[6] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-Scale Simple Question Answering with Memory Networks. *arXiv preprint arXiv:1506.02075* (2015).

[7] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *NIPS*. 2787–2795.

[8] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning Structured Embeddings of Knowledge Bases. In *AAAI*.

[9] Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014. Open Question Answering with Weakly Supervised Embedding Models. In *ECML PKDD*. 165–180.

[10] Zihang Dai, Lei Li, and Wei Xu. 2016. CFO: Conditional Focused Neural Question Answering with Large-Scale Knowledge Bases. *arXiv preprint arXiv:1606.01994* (2016).

[11] Rajarshi Das, Manzil Zaheer, Siva Reddy, and Andrew McCallum. 2017. Question Answering on Knowledge Bases and Text using Universal Schema and Memory Networks. In *ACL*.

[12] Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question Answering Over Freebase With Multi-Column Convolutional Neural Networks. In *ACL-IJCNLP*. 260–269.

[13] Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-Driven Learning for Open Question Answering. In *ACL*. 1608–1618.

[14] Miao Fan, Kai Cao, Yifan He, and Ralph Grishman. 2015. Jointly Embedding Relations and Mentions for Knowledge Population. In *RANLP*. 186–191.

[15] Miao Fan, Yue Feng, Mingming Sun, Ping Li, Haifeng Wang, and Jianmin Wang. 2018. Multi-Task Neural Learning Architecture for End-to-End Identification of Helpful Reviews. In *ASONAM*. 343–350.

[16] Miao Fan, Wutao Lin, Yue Feng, Mingming Sun, and Ping Li. 2018. A Globalization-Semantic Matching Neural Network for Paraphrase Identification. In *CIKM*. 2067–2075.

[17] Miao Fan, Qiang Zhou, Thomas Fang Zheng, and Ralph Grishman. 2017. Distributed Representation Learning for Knowledge Graphs with Entity Descriptions. *Pattern Recognition Letters* 93 (2017), 31–37.

[18] David Golub and Xiaodong He. 2016. Character-Level Question Answering with Attention. In *EMNLP*. 1598–1607.

[19] Google. 2018. Freebase Data Dumps. https://developers.google.com/freebase.

[20] Dilek Hakkani-Tür, Asli Celikyilmaz, Larry Heck, Gokhan Tur, and Geoff Zweig. 2014. Probabilistic Enrichment of Knowledge Graph Entities for Relation Detection in Conversational Understanding. In *INTERSPEECH*.

[21] Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao. 2017. An End-to-End Model for Question Answering over Knowledge Base with Cross-Attention Combining Global Knowledge. In *ACL*. 221–231.

[22] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. DBpedia–A Large-Scale, Multilingual Knowledge Base Extracted From Wikipedia. *Semantic Web* 6, 2 (2015), 167–195.

[23] Dingcheng Li, Jingyuan Zhang, and Ping Li. 2018. Representation Learning for Question Classification via Topic Sparse Autoencoder and Entity Embedding. In *IEEE Big Data*.

[24] Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015. Modeling Relation Paths for Representation Learning of Knowledge Bases. In *EMNLP*. 705–714.

[25] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *AAAI*. 2181–2187.

[26] Teng Long, Ryan Lowe, Jackie Chi Kit Cheung, and Doina Precup. 2016. Leveraging Lexical Resources for Learning Entity Embeddings in Multi-Relational Data. In *ACL*. 112–117.

[27] Denis Lukovnikov, Asja Fischer, Jens Lehmann, and Sören Auer. 2017. Neural Network-Based Question Answering over Knowledge Graphs on Word and Character Level. In *WWW*. 1211–1220.

[28] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality. In *NIPS*. 3111–3119.

[29] Salman Mohammed, Peng Shi, and Jimmy Lin. 2018. Strong Baselines for Simple Question Answering over Knowledge Graphs with and without Neural Networks. In *NAACL-HLT*. 291–296. https://github.com/castorini/BuboQA

[30] Aasish Pappu, Roi Blanco, Yashar Mehdad, Amanda Stent, and Kapil Thadani. 2017. Lightweight Multilingual Entity Extraction and Linking. In *WSDM*. 365–374.

[31] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *EMNLP*. 1532–1543.

[32] Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning with Neural Tensor Networks for Knowledge Base Completion. In *NIPS*. 926–934.

[33] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. YAGO: A Core of Semantic Knowledge. In *WWW*. 697–706.

[34] Yi Tay, Anh Tuan Luu, Siu Cheung Hui, and Falk Brauer. 2017. Random Semantic Tensor Ensemble for Scalable Knowledge Graph Link Prediction. In *WSDM*. 751–760.

[35] Ferhan Ture and Oliver Jojic. 2017. No Need to Pay Attention: Simple Recurrent Neural Networks Work!. In *EMNLP*. 2866–2872.

[36] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge Graph Embedding: A Survey of Approaches and Applications. *TKDE* 29, 12 (2017), 2724–2743.

[37] Zhigang Wang and Juanzi Li. 2016. Text-Enhanced Representation Learning for Knowledge Graph. In *IJCAI*. 1293–1299.

[38] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph and Text Jointly Embedding. In *EMNLP*. 1591–1601.

[39] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. In *AAAI*.

[40] Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. 2013. Connecting Language and Knowledge Bases with Embedding Models for Relation Extraction. In *EMNLP*. 1366–1371.

[41] Han Xiao, Minlie Huang, Lian Meng, and Xiaoyan Zhu. 2017. SSP: Semantic Space Projection for Knowledge Graph Embedding with Text Descriptions. In *AAAI*. 3104–3110.

[42] Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation Learning of Knowledge Graphs with Entity Descriptions. In *AAAI*. 2659–2665.

[43] Min-Chul Yang, Nan Duan, Ming Zhou, and Hae-Chang Rim. 2014. Joint Relational Embeddings for Knowledge-Based Question Answering. In *EMNLP*. 645–650.

[44] Min-Chul Yang, Do-Gil Lee, So-Young Park, and Hae-Chang Rim. 2015. Knowledge-Based Question Answering Using the Semantic Embedding Space. *Expert Systems with Applications* 42, 23 (2015), 9086–9104.

[45] Scott Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base. In *ACL-IJCNLP*.

[46] Wenpeng Yin, Mo Yu, Bing Xiang, Bowen Zhou, and Hinrich Schütze. 2016. Simple Question Answering by Attentive Convolutional Neural Network. In *COLING*. 1746–1756.

[47] Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2017. Improved Neural Relation Detection for Knowledge Base Question Answering. In *ACL*. 571–581.

[48] Dongxu Zhang, Bin Yuan, Dong Wang, and Rong Liu. 2015. Joint Semantic Relevance Learning with Text Data and Graph Knowledge. In *Workshop on Continuous Vector Space Models and their Compositionality*. 32–40.

[49] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative Knowledge Base Embedding for Recommender Systems. In *KDD*. 353–362.

[50] Huaping Zhong, Jianwen Zhang, Zhen Wang, Hai Wan, and Zheng Chen. 2015. Aligning Knowledge and Text Embeddings by Entity Descriptions. In *EMNLP*. 267–272.