Knowledge Base Question Answering with Topic Units

Yunshi Lan , **Shuohang Wang** and **Jing Jiang**
School of Information System, Singapore Management University
{yslan.2015, shwang.2014}@phdis.smu.edu.sg, jingjiang@smu.edu.sg

Abstract

Knowledge base question answering (KBQA) is an important task in natural language processing. Existing methods for KBQA usually start with entity linking, which considers mostly named entities found in a question as the starting points in the KB to search for answers to the question. However, relying only on entity linking to look for answer candidates may not be sufficient. In this paper, we propose to perform topic unit linking where topic units cover a wider range of units of a KB. We use a generation-and-scoring approach to gradually refine the set of topic units. Furthermore, we use reinforcement learning to jointly learn the parameters for topic unit linking and answer candidate ranking in an end-to-end manner. Experiments on three commonly used benchmark datasets show that our method consistently works well and outperforms the previous state of the art on two datasets.

1 Introduction

Knowledge base question answering (KBQA) is an important task in NLP that has many real-world applications such as in search engines and decision support systems. It has attracted much attention in recent years [Cai and Yates, 2013; Kwiatkowski *et al.*, 2013; Bordes *et al.*, 2014; Dong *et al.*, 2015; Yih *et al.*, 2015; Xu *et al.*, 2016; Yu *et al.*, 2017; Petrochuk and Zettlemoyer, 2018]. The task is defined as finding the answer to a factoid question using facts stored in a knowledge base (KB).

Most existing methods for KBQA use a pipelined approach: First, given a question q, an *entity linking* step is used to find KB entities mentioned in q. These entities are often referred to as *topic entities*. Next, relations or relation paths in the KB linked to the topic entities are ranked such that the best relation or relation path matching q is selected as the one that leads to the answer entities. For example, given the question Q1 shown in Figure 1, an entity linking tool may link the phrase "Morgan Freeman" in the question to the entity "Morgan Freeman" in the KB. Then starting from this topic entity, a number of different relation paths are considered such as (Morgan Freeman, education), (Morgan Freeman, place of birth) and (Morgan Freeman, place of birth, contained by).

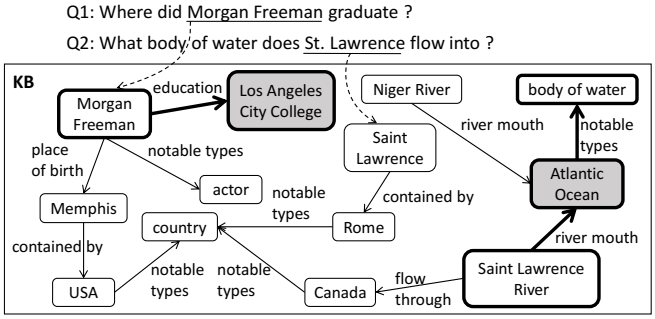


Figure 1: Two example questions and how they can be answered by a KB. The questions are linked to *topic entities* by imaginary lines. The shaded entities are the correct answers to the questions. The paths in bold are correct relation paths towards the questions.

Ideally, we want the path (Morgan Freeman, education) to be ranked the first with respect to the question so that the correct answer at the end of this path can be extracted.

Although a plethora of methods has been proposed for KBQA, most work focuses on the relation path ranking step. For entity linking, many methods rely entirely on existing entity linking tools [Xu *et al.*, 2016; Dong *et al.*, 2017; Hao *et al.*, 2017], which generally use traditional rule-based methods to perform named entity recognition and linking. There are at least two limitations with this approach. First, oftentimes an entity mention in a question is ambiguous and an entity linking tool may not link it to the correct entity in the KB. Take the question Q2 in Figure 1 for example. An entity linking tool is more likely to mistakenly link the entity mention "St. Lawrence" to the entity "Saint Lawrence" in the KB, which is far away from the correct answer entity "Atlantic Ocean." On the other hand, the question in Figure 1 shows that words that are not part of a named entity, such as "body", "water" and "flow", can also be linked to relevant entities and relations in the KB such as "body of water" and "flow through" that can help find the correct answer entity. The second limitation with a pipeline approach is that the entity linking step cannot be trained using the final KBQA results. Again, let us look at the Q2 in Figure 1. If both "Saint Lawrence" and "Saint Lawrence River" are recognized as topic entities, an entity linking module developed outside of the KBQA system would not be able to know which one

is more relevant to the question. However, if we could train a topic entity ranking function using the ground truth answer to the question, we may learn that with "water" and "flow" appearing in the question, "Saint Lawrence River" should be ranked higher than "Saint Lawrence" as a topic entity.

In this paper, we address the two limitations above by replacing the standard topic entity linking module with a novel *topic unit generation-and-scoring* module. Our topic units include not only named entities but also other KB units such as entities containing common nouns (e.g., "body of water") and relation types (e.g., "flow through," "river mouth"). By flexibly considering a wide range of topic units, we can increase the chance of the correct answer being connected to one of the topic units. However, we do not want to consider too many topic units for the subsequent relation path ranking step as this would incur high computational costs. We therefore propose to identify topic units in two steps: a generation step and a scoring step. First, in a topic unit generation step, we use heuristics with low computational costs (e.g., n-gram matching with an inverted index) to identify an initial set of topic units that has a high coverage. Subsequently, in a topic unit scoring step, we use a neural network-based scoring function to rank the initial topic units and select a small number of them that are highly relevant to the question. We then only consider relation paths derived from this small set of topic units. Our method is trained in an end-to-end manner using reinforcement learning such that the ranking function in the topic unit scoring step can be learned without knowing the ground truth topic units.

We evaluate our method on three benchmark datasets: WebQuestionsSP, ComplexWebQuestions and SimpleQuestions. We find that our method can clearly outperform the state of the art on two datasets, especially on ComplexWebQuestions where the improvement is substantial. It also performs competitively on the third dataset. Further analyses also show that considering a wide range of topic units is crucial to the performance improvement.

2 Related Work

A general solution to KBQA is a pipeline approach, where the question is first linked to some topic entities in the KB by an existing entity linking tool [Yao and Durme, 2014; Bordes *et al.*, 2015; Dong *et al.*, 2017; Hao *et al.*, 2017] and then relation paths connected to the topic entities in the KB are retrieved and ranked by various ranking models, including semantic parsing based models [Berant *et al.*, 2013; Yih *et al.*, 2015; Yih *et al.*, 2016] and embedding based models [Bordes *et al.*, 2014; Bordes *et al.*, 2015; Dong *et al.*, 2017; Sun *et al.*, 2018]. The top-ranked relation path will then be used to answer the question. Although these methods have worked well on KBQA, there are limitations of this pipeline approach using an external entity linking tool [Shen *et al.*, 2015; Lample *et al.*, 2016; Mai *et al.*, 2018], as we have pointed out in Section 1.

There has been some work attempting to address the limitations pointed out. Yu *et al.* (2017) re-rank the topic entities using their associated relations. Xu *et al.* (2016) jointly train entity linking and path ranking through entity descriptions. However, these studies still only consider named entities as topic entities rather than linking a question to other kinds of KB units. In contrast, we consider a wide range of KB units during our topic unit linking step. Zhang *et al.* (2017) integrate entity linking and relation path ranking into an end-to-end model. However, their topic entity linking module considers all candidate topic entities without using a scoring function to rank and fiter them, and as a result, it is hard to scale up their method to large knowledge bases. In contrast, we take a topic unit generation-and-scoring approach, using heuristics for generating topic units and a neural network model for fine-grained scoring. Similar to [Zhang *et al.*, 2017], We also use reinforcement learning to jointly train the entire system.

3 Method

3.1 Task Setup

We first formally define the KBQA task. A KB (knowledge base) consists of a set of entities $\mathcal{E}$ (e.g., *Morgan Freeman*), a set of relation types[1] $\mathcal{R}$ (e.g., *place of birth*) and a set of relation triplets (h, r, t) (e.g., $(\textit{Morgan Freeman}, \textit{place of birth}, \textit{Memphis})$, where $h \in \mathcal{E}$ is the head entity, $t \in \mathcal{E}$ is the tail entity, and $r \in \mathcal{R}$ is a directed relation between h and t. The triplets (h, r, t) are facts or knowledge contained in the KB. A KB can also be seen as a *knowledge graph* whose nodes are the entities and edges are the relations. We assume that each entity or relation type has a textual description, which is a sequence of words.

We define *KB units* (denoted as $\mathcal{U}$) to be the union of the entities and the relation types, i.e., $\mathcal{U} = \mathcal{E} \cup \mathcal{R}$. Given a question q, the task of KBQA (knowledge base question answering) is to find a set of entities in $\mathcal{E}$ that are answers to q. It is assumed that a set of questions together with their correct answers in the KB is given for training.

3.2 Method Overview

Like many existing methods for KBQA, our method follows the general approach of first linking the words in the question to some parts of the KB as starting points for search and then following the edges of the knowledge graph to look for answer entities whose relation path best matches the question. Different from previous methods, instead of confining the first linking step to only named entities in the question, we consider a wider range of KB units to be linked to the question. We also propose a generation-and-scoring strategy such that we can gradually refine the linked KB units.

Specifically, we divide our method into the following three steps: (1) **Topic unit generation**. In this step, our goal is to identify all KB units that are likely mentioned in the question. We want to achieve high coverage without incurring any heavy computation. So in this step we rely mostly on existing entity linking tools as well as string matching, pre-computed corpus statistics and pre-constructed inverted index. The output of this step is an initial set of topic units for a given question. (2) **Topic unit scoring**. In this second step, we want

[1] Since sometimes *relations* may refer to relation triplets, to avoid confusion, here we use the term *relation type*.

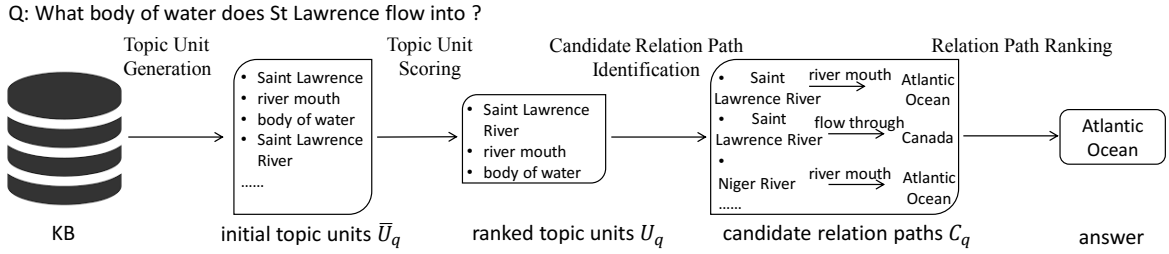


Figure 2: An overview of the various steps of our method.

to refine the topic units obtained in the first step by selecting the top ones based on a sophisticated ranking function learned from the training data. The output of this step is a much smaller subset of the initial topic units. (3) **Relation path ranking**. Given the topic units selected in the previous step, we can derive a set of relation paths where each path starts from a topic unit and contains one or multiple hops of relations. We then use a neural network-based scoring function to rank these relation paths. Finally, we pick the entities connected to the top-ranked relation path as the answers to the question.

Our main contributions lie in the first two steps. We regard the third step as a standard procedure to complete the entire KBQA system. The three steps are illustrated in Figure 2. In the rest of this section we present the details of each step and describe how we use reinforcement learning to train the entire method in an end-to-end manner.

3.3 Topic Unit Generation

The goal of topic unit generation is to identify all KB units that are possibly mentioned in the question. For named entities appearing in the question, we rely on existing entity linking tools to recognize and link them. Here we mainly describe how we identify other KB units possibly mentioned in the question.

A straightforward solution would be to identify those KB units whose textual descriptions contain one of the words in the question. However, exact word matching is too restrictive. Here we use two strategies to relax the matching conditions. One is to allow character-level n-gram matching. Another is to expand the question with additional words that are highly related to some original question words.

Specifically, we do the following to identify topic units that are not named entities:

1. We build an inverted index to map each unique character n-gram (where $n > 4$) and each unique word found in the descriptions of all KB units to the corresponding KB units. This allows us to quickly link a character n-gram or a word found in a question to the KB units contain it.
2. Next, we link the question to some highly correlated words based on statistics obtained from the training data. E.g., the word "flow" appearing in the question in Figure 2 could be linked to the word "river" in a relation path, which would help us link "flow" from a question to relation types such as "river mouth" in the KB. To achieve this, for those training questions whose topic entities can be identified by existing entity linking tools, we find the relation paths between the topic entities and the ground truth answer entities. We thus obtain a set of (question, relation path) pairs. We then compute the pointwise mutual information (PMI) between each pair of a question word and a relation path word. Note that although the relation paths used here are not always correct, most of them are still relevant to the question, and therefore the mutual information computed can still indicate strongly correlated words.
3. Given a question q, we first use an entity linking tool to identify named entities in q. Then we remove the linked named entities and stop words in q. For the remaining question words, we use the pre-computed PMI values to find other words highly correlated with one of the question words (using a PMI threshold of 1) and add these additional words to the question.
4. Finally, we find those KB units linked to all the character n-grams and the words inside the expanded question, based on the inverted index built earlier. This is our initial set of topic units for question q, which we denote with $\bar{\mathcal{U}}_q$.

3.4 Topic Unit Scoring

The topic unit generating step aims to increase coverage but usually returns a large set of topic units. In the topic unit scoring step, we use a scoring function to derive a distribution over the topic units identified from the previous step with respect to the question.

The probability function is based on a standard linear feed-forward neural network as follows:

$$\begin{aligned} s(u,q) &= \mathbf{w}_1^{\mathsf{T}}\mathbf{f}_u + b_1, \\ p(u|q) &= \frac{\exp(s(u,q))}{\sum_{u' \in \bar{\mathcal{U}}_q} \exp(s(u',q))}, \end{aligned} \tag{1}$$

where $u \in \bar{\mathcal{U}}_q$, $\mathbf{f}_u$ is a feature vector associated with u, and $\mathbf{w}_1$ and b_1 are parameters to be learned.

The feature vector $\mathbf{f}_u$ is the concatenation of four vectors:

$$\mathbf{f}_u = \mathbf{f}_u^{\text{semantic}} \oplus \mathbf{f}_u^{\text{character}} \oplus \mathbf{f}_u^{\text{category}} \oplus \mathbf{f}_u^{\text{link}}.$$

- $\mathbf{f}_u^{\text{semantic}}$ is the output of a neural network-based sequence matching model [Seo *et al.*, 2017] that measures the semantic relatedness between the topic unit u and the question q. Here the topic unit u is a represented by the embedding vectors of the words in the textual description of u, denoted as $(\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{|u|})$, and q is also

represented by the embedding vectors of its sequence of words, $(\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_{|q|})$. The sequence matching model first uses the attention mechanism to obtain attention weights a_{ij} as follows:

$$\begin{aligned} \alpha_{ij} &= \mathbf{w}_2^\intercal(\mathbf{q}_i \oplus \mathbf{u}_j \oplus (\mathbf{q}_i \odot \mathbf{u}_j)), \\ a_{ij} &= \frac{\exp(\alpha_{ij})}{\sum_{k=1}^{|q|}\exp(\alpha_{kj})}. \end{aligned}$$

Then for each word $\mathbf{u}_j$ in the topic unit u, we obtain an attention-weighted version of the question as follows:

$$\tilde{\mathbf{q}}_j = \sum_{i=1}^{|q|} a_{ij}\mathbf{q}_i.$$

Finally, we match each unit word $\mathbf{u}_j$ with its corresponding question representation $\tilde{\mathbf{q}}_j$, and aggregate the matching results to obtain the vector $\mathbf{f}_u^{\text{semantic}}$:

$$\mathbf{f}_u^{\text{semantic}} = \mathbf{w}_3^\intercal \sum_{j=1}^{|u|}(\tilde{\mathbf{q}}_j \oplus \mathbf{u}_j).$$

Essentially $\mathbf{f}_u^{\text{semantic}}$ is a 1-dimensional vector that encodes the knowledge about how well unit u matches question q using the attention-based sequence matching model. Here $\mathbf{w}_2$ and $\mathbf{w}_3$ are parameters to be learned.

- $\mathbf{f}_u^{\text{character}}$ is a 1-dimensional vector that measures the percentage of characters in u that are also found in q.
- $\mathbf{f}_u^{\text{category}}$ is a 3-dimensional one-hot vector indicating the category of the topic unit u, i.e., whether it is a named entity recognized by the entity linking tool, an entity that is a common noun, or a relation type.
- $\mathbf{f}_u^{\text{link}}$ is a 1-dimensional vector, which contains the entity linking score returned by the entity linking tool if u is a named entity and 0 otherwise.

Based on Eqn. (1), we can rank the topic units in $\bar{\mathcal{U}}_q$ and pick the top-K to form a new set $\mathcal{U}_q$ for the next step.

3.5 Relation Path Ranking

Given the set $\mathcal{U}_q$, in the relation path ranking step we first identify candidate relation paths that are connected to some $u \in \mathcal{U}_q$ and then rank these relation paths based on how well they match the question. Note that this step is not the focus of our work and we omit some of the implementation details.

To identify the candidate relation paths, for each $u \in \mathcal{U}_q$ we extract a set of relation paths. If u is an entity, we take those relation paths starting from u and containing one or two relations. If u itself is a relation type, we take all relation paths in the KB with one or two relations where at least one of the relations is u. When a unit u gives us more than 500 relation paths, we use character-level overlap with the question q to rank these relation paths and take only the top 500. In the end, we take the union of all the extracted relation paths from all $u \in \mathcal{U}_q$. Let us use $\mathcal{C}_q$ to denote this set of candidate relation paths.

Next, we would like to obtain a distribution over the paths inside $\mathcal{C}_q$. We again use a standard linear feed-forward neural network for this:

$$\begin{aligned} s(c, q) &= \mathbf{w}_4^\intercal \mathbf{f}_c + b_4, \\ p(c|q) &= \frac{\exp(s(c, q))}{\sum_{c' \in \mathcal{C}_q}\exp(s(c', q))}, \end{aligned} \tag{2}$$

where $c \in \mathcal{C}_q$, $\mathbf{w}_4$ and b_4 are parameters to be learned, and

$$\mathbf{f}_c = \mathbf{f}_c^{\text{link}} \oplus \mathbf{f}_c^{\text{semantic}} \oplus \mathbf{f}_c^{\text{pattern}} \oplus \mathbf{f}_c^{\text{answer}}.$$

The four feature vectors are defined as follows:

- For a candidate path c, we consider all its components that are topic units inside $\mathcal{U}_q$ and define $\mathbf{f}_c^{\text{link}}$ to be a 1-dimensional vector containing the sum of the probabilities of these units as computed by Eqn. (1).
- $\mathbf{f}_c^{\text{semantic}}$ is based on a previous work on KBQA [Yih *et al.*, 2015]. It is the output of a sequence matching model that matches the question sequence with the relation path sequence, where the relation path sequence contains the words from the descriptions of the components of the relation paths and a CNN network with a max-pooling layer is used to encode each sequence into a vector before the dot product of the two vectors is computed to measure their similarity.
- $\mathbf{f}_c^{\text{pattern}}$ is also based on some existing work on KBQA [Dong *et al.*, 2015]. It also uses the same sequence matching model [Yih *et al.*, 2015] to measure the similarity between the question and a relation path, but entities in the question and the relation path are replaced by placeholders. E.g., "where did Morgan Freeman graduate" becomes (where, did, ⟨e⟩, graduate).
- Since previous work [Xu *et al.*, 2016] has shown that contexts in the KB of candidate answer entities are useful for KBQA, here we adopt this idea to define $\mathbf{f}_c^{\text{answer}}$. For the relation path c, we collect those entities linked to the answer entities that characterize them. E.g., from the answer entity "Jamie Dornan" we can collect entities "person" and "actor," which are linked to "Jamie Dornan" in the KB. We call these answer contexts. We then use another CNN network to match the question pattern with these answer contexts to derive $\mathbf{f}_c^{\text{answer}}$.

3.6 End-to-End Learning

The model parameters we need to learn include those used in the scoring function at the topic unit scoring step (which we denote as θ_1) and those in the scoring function at the relation path ranking step (which we denote as θ_2). Although these are two separate steps, we jointly learn the parameters in an end-to-end manner. (See Algorithm 1.)

We define the overall loss function as follows. For each relation path, we treat the set of the connected tail entities as the predicted answer. Given a training question q and its ground truth answers, we can compute the F1 score of each relation path $c \in \mathcal{C}_q$. We normalize these F1 scores over all paths and treat it as an empirical distribution over $\mathcal{C}_q$, which we denote as $\hat{p}(c|q)$. We use the KL-divergence between $\hat{p}(c|q)$ and

Algorithm 1 Model training

Input: *KB*, training questions $\mathcal{Q}$ and their answers
Output: (θ_1, θ_2)
Initialize: $(\theta_1, \theta_2) \leftarrow$ pre-trained models (see Section 3.7)
for each $q \in \mathcal{Q}$ **do**
    Identify the initial set $\bar{\mathcal{U}}_q$ according to Section 3.3
    Sample K topic units $\hat{\mathcal{U}}_q$ according to $p_{\theta_1}(u|q)$
    Use the topic units $\hat{\mathcal{U}}_q$ to identify relation paths $\mathcal{C}_q$
    Rank the relation paths using $p_{\theta_2}(c|q)$ and pick the top one to extract the answers
    Compute the reward $r(\hat{\mathcal{U}}_q)$ based on the F1 score of the answers
    Update θ_1 through the policy gradient according to Eqn. (4)
    Update θ_2 through the gradient of the loss according to Eqn. (3)

the predicted distribution $p(c|q)$ from Eqn. (2) to measure the loss on q, and we sum over all the training questions in our training set $\mathcal{Q}$ as the total loss:

$$L(\theta_1, \theta_2, \mathcal{Q}) = -\sum_{q \in \mathcal{Q}} \sum_{c \in \mathcal{C}_q} \hat{p}(c|q) \log \frac{p(c|q)}{\hat{p}(c|q)}. \quad (3)$$

Because at the topic unit scoring step we pick the top-K topic units, which is a discrete choice, the loss function above is not differentiable over θ_1. We thus adopt reinforcement learning and use policy gradient to learn θ_1 [Williams, 1992]. Specifically, let $r(\hat{\mathcal{U}}_q)$ denote the reward of selecting a set of K topic units $\hat{\mathcal{U}}_q \subset \bar{\mathcal{U}}_q$ as the final topic units, the gradient for θ_1 is

$$\nabla_{\theta_1} J(\theta_1) = \mathbb{E}[r(\bar{\mathcal{U}}_q) \cdot \nabla_{\theta_1} \log p_{\theta_1}(u|q)]. \quad (4)$$

For the reward $r(\bar{\mathcal{U}}_q)$, we use the final F1 score of the extracted answers when $\bar{\mathcal{U}}_q$ is selected. We use the sampling method to estimate the expected reward. To update θ_2, we fix θ_1 and use the loss function shown in Eqn. (3).

3.7 Implementation Details

We use S-MART [Yang and Chang, 2015] as our entity linking tool. We leverage 300-dimensional GloVe [Pennington *et al.*, 2014] word embeddings at the topic unit scoring step and the relation path ranking step. We use Adam optimizer with an initial learning rate of 0.001. All hidden vectors are 200-dimensional. All hyper-parameters are turned on the development data. During training of reinforcement learning, to initialize θ_1, we distantly train the model with surrogate labels of the topic units by checking whether the unit has a relation path leading to the correct answer. To initialize θ_2, we train a baseline using only topic entities returned by our entity linking tool. We set K to be 3.

4 Experiments

4.1 Datasets

We evaluate our KBQA method on three benchmark datasets. **WebQuestionsSP (WQSP)**: This is a dataset that has been widely used for KBQA [Yih *et al.*, 2016]. It contains 2848 training questions, 250 development questions and 1639 test questions. **ComplexWebQuestions (CWQ)**: This dataset was introduced by Talmor and Berant [2018] with the intention to create more complex questions from the WebQuestionsSQ dataset. Questions in this dataset often involve relation paths with more than one relations. CWQ contains 27K, 3K and 3K questions for training, development and test, respectively. **SimpleQuestions (SQ)**: This is another popularly used KBQA dataset, introduced by Bordes *et al.* [2015]. Questions in this dataset can be answered by single-hop relation paths. SQ contains 76K, 11K and 21K for training, development and test, respectively.

For WQSP and CWQ, the knowledge base used is the entire Freebase. For SQ, the knowledge base used is a subset of Freebase that comes with the SQ dataset, which is called "FB2M." To measure the performance, we follow the standard evaluation metric for each dataset. We use hits@1 and F1 scores for WQSP and CWQ, and accuracy for SQ[2].

4.2 Main Results

First, we would like to check whether our ideas to consider a wide range of KB units for topic unit linking and to use the generation-and-scoring strategy work. We use ablation experiments to do the comparison. In Table 2, **FullModel** refers to our full model that first links a question to a wide range of KB units and then uses the topic unit scoring function to select a small set of topic units for the subsequent relation path ranking. **NEOnly** refers to a degenerate version of the full model where during the topic unit generation step we only consider named entities, i.e., we only use the topic entities returned by the entity linking tool. However, the topic unit scoring step is still retained and trained using reinforcement learning. **BL** refers to a baseline version of our method where only named entities are considered as topic units (same as in **NEOnly**) and there is no scoring and selection of these topic entities.

From Table 2 we have the following findings: (1) **FullModel** consistently works better than **NEOnly** on all three datasets, verifying the effectiveness of including a wide range of KB units as topic units in the topic unit generation step. Note that since both **FullModel** and **NEOnly** have the topic unit scoring step, their performance difference is not due to the neural network-based ranking function. (2) **NEOnly** consistently works better than **BL** on all three datasets, showing that even with only named entities as topic units, it is still beneficial to use the neural network-based ranking function to select the top topic units for the subsequent relation path ranking. (3) The improvement of **NEOnly** over **BL** is not as large as the improvement of **FullModel** over **NEOnly**, suggesting that the idea of using a wide range of KB units for topic unit linking is more important.

4.3 Comparison with Existing Methods

Next, we compare our method with the state-of-the-art (SOTA) performance on each of the three datasets.

[2]For hits@1, we used the official evaluation script at https://www.tau-nlp.org/compwebq. For F1, we retrieved the ground truth via SPARQL queries and measured by ourselves.

Method	WQSP	CWQ	SQ
Our FullModel	**68.2** / 67.9	**39.3 / 36.5**	**80.3**
SOTA	-/**69.0**	34.2/-	78.1
HR-BiLSTM	- / -	- / -	77.0
GRAFT-Net	67.8 / 62.8	- / -	-
HR-BiLSTM†	62.9 / 62.3	33.3 / 31.2	77.6
GRAFT-Net†	67.8 / 62.5	30.1 / 26.0	-

(a)

		WQSP	CWQ	SQ
avg size of $\bar{\mathcal{U}}_q$	FullModel	24.5	22.6	194.0
	NEOnly	2.9	9.8	163.6
topic unit recall	FullModel	97.3%	83.1%	97.8%
	NEOnly	93.3%	78.4%	96.7%
answer recall	FullModel	93.4%	52.0%	97.7%
	NEOnly	89.5%	48.0%	96.6%

(b)

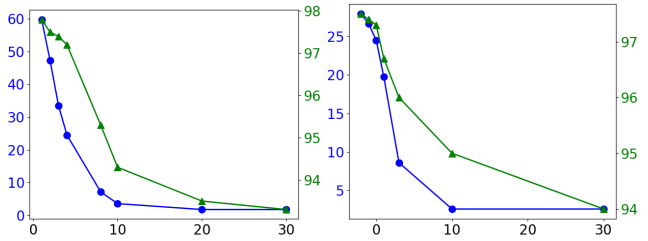
Feature	WQSP
all features	67.9
without $\mathbf{f}^{\text{semantic}}$	61.5 (-6.4)
without $\mathbf{f}^{\text{character}}$	65.9 (-2.0)
without $\mathbf{f}^{\text{category}}$	66.1 (-1.8)
without $\mathbf{f}^{\text{link}}$	64.9 (-3.0)

(c)

Table 1: (a) Comparison with existing methods. The top section shows the performance of our full model. The middle section shows previously reported performance. The last section shows the performance of two existing methods reimplemented by us. (b) Coverage of $\bar{\mathcal{U}}_q$. (c) F1 scores on WQSP when different feature configurations are used in Eqn. (1).

Method	WQSP	CWQ	SQ
FullModel	**68.2/67.9**	**39.3/36.5**	**80.3**
NEOnly	64.8/64.0	38.4/34.0	78.2
BL	63.6/62.8	36.3/33.8	77.9

Table 2: The main experiment results. The metrics used are the commonly-used ones for each dataset. For WQSP and CWQ the metrics are hits@1/F1. For SQ the metrics are accuracy.



(a) (b)

Figure 3: Performance on WQSP test in terms of avg size of $\bar{\mathcal{U}}_q$ (in blue) and topic unit recall (in green) when (a) n-gram threshold ranges from 0 to 30 and PMI threshold is 1. (b) n-gram threshold is 4 and PMI threshold ranges from -2 to 30.

NSM [Chen Liang, 2017], SPLITQA [Talmor and Berant, 2018] and BiLSTM-CRF [Petrochuk and Zettlemoyer, 2018] achieve the state of the art on WQSP, CWQ and SQ, respectively. We show their originally reported results in Table 1a. Besides, we also consider two recent methods that have been shown to generally work well for KBQA: HR-BiLSTM [Yu *et al.*, 2017] and GRAFT-Net [Sun *et al.*, 2018]. We reimplemented these two methods and report both our results and the originally reported results of these two methods.

From Table 1a we can see the following: (1) Our full model outperforms the previous state of the art on CWQ and SQ. On WQSP, in terms of hits@1, our model also achieves the state of the art, while in terms of F1, our model still performs competitively although not as good as NSM. (2) Previous state-of-the-art methods NSM, SPLITQA and BiLSTM-CRF were each tested on a single dataset. It is unclear whether they could perform consistently well on different datasets. Our full model is shown to consistently work well on three datasets.

4.4 Further Analyses

Coverage of $\bar{\mathcal{U}}_q$. We would like to check if the initial set of topic units $\bar{\mathcal{U}}_q$ as returned by our method indeed has a higher coverage than traditional entity linking. We therefore compare **FullModel** and **NEOnly** in three aspects. We first look at the average size of $\bar{\mathcal{U}}_q$. We then look at *topic unit recall* of $\bar{\mathcal{U}}_q$. This is defined as the percentage of questions for which $\bar{\mathcal{U}}_q$ contains at least one of the KB units found in the ground truth relation paths (which are provided in the datasets but not used for training in our method). We also look at *answer recall* of $\bar{\mathcal{U}}_q$, which is defined as the percentage of questions for which one of the relation paths derived from $\bar{\mathcal{U}}_q$ leads to the correct answer. We show the numbers in Table 1b. We can see that indeed FullModel gives a larger size of $\bar{\mathcal{U}}_q$ in general and can increase the topic unit recall and answer recall.

Configurations of Topic Unit Generation. In the topic unit generation step, we defined some thresholds, namely, n-gram threshold and PMI threshold. Figure 3 shows that with increasing n-gram and PMI thresholds, both average size of $\bar{\mathcal{U}}_q$ and topic unit recall decrease. To obtain scalable topic units without too much decrease of topic unit recall, we set n-gram and PMI thresholds as 4 and 1, respectively.

Features for Topic Unit Scoring. Recall that in the topic unit scoring step our scoring function uses four feature vectors. In Table 1c we show the performance in terms of F1 on the WQSP dataset when we use the full model and when we remove each of the feature vectors. We can see that if we remove any of the feature vectors, the performance drops. In particular, the performance decreases the most when the feature $\mathbf{f}^{\text{semantic}}$ is removed, showing the importance of measuring the semantic relevance of a topic unit to the question.

5 Conclusions

In this paper we propose method that uses topic units for KBQA, which allows us to leverage more information of the questions. We show that our method can achieve either the state of the art or competitive results on benchmark datasets.

Acknowledgements

This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its International Research Centres in Singapore Funding Initiative.

References

[Berant *et al.*, 2013] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of EMNLP*, pages 1533–1544, 2013.

[Bordes *et al.*, 2014] Antoine Bordes, Sumit Chopra, and Jason Weston. Question answering with subgraph embeddings. In *Proceedings of EMNLP*, pages 615–620, 2014.

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

[Cai and Yates, 2013] Qingqing Cai and Alexander Yates. Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of ACL*, pages 423–433, 2013.

[Chen Liang, 2017] Quoc Le Kenneth D. Forbus Ni Lao Chen Liang, Jonathan Berant†. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of ACL*, pages 23–33, 2017.

[Dong *et al.*, 2015] Li Dong, Furu Wei, Ming Zhou, and Ke Xu. Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of ACL-IJCNLP*, pages 260–269, 2015.

[Dong *et al.*, 2017] Li Dong, Jonathan Mallinson, Siva Reddy, and Mirella Lapata. Learning to paraphrase for question answering. In *Proceedings of EMNLP*, pages 875–886, 2017.

[Hao *et al.*, 2017] Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhiyuan Liu, hua Wu, and Jun Zhao. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In *Proceedings of ACL*, pages 221–231, 2017.

[Kwiatkowski *et al.*, 2013] Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. Scaling semantic parsers with on-th-fly ontology matching. In *Proceedings of EMNLP*, pages 1545–1556, 2013.

[Lample *et al.*, 2016] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*, pages 260–270, 2016.

[Mai *et al.*, 2018] Khai Mai, Thai-Hoang Pham, Minh Trung Nguyen, Tuan Duc Nguyen, Danushka Bollegala, Ryohei Sasano, and Satoshi Sekine. An empirical study on fine-grained named entity recognition. In *Proceedings of COLING*, pages 711–722, 2018.

[Pennington *et al.*, 2014] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543, 2014.

[Petrochuk and Zettlemoyer, 2018] Michael Petrochuk and Luke Zettlemoyer. Simple questions nearly solved: a new upperbound and baseline approach. In *Proceedings of EMNLP*, pages 554–558, 2018.

[Seo *et al.*, 2017] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. In *Proceedings of ICLR*, 2017.

[Shen *et al.*, 2015] Wei Shen, Jianyong Wang, and Jiawei Han. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE TKDE*, 27(2):443–460, 2015.

[Sun *et al.*, 2018] Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William W. Cohen. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of EMNLP*, pages 4231–4242, 2018.

[Talmor and Berant, 2018] Alon Talmor and Jonathan Berant. The web as a knowledge-base for answering complex questions. In *Proceedings of NAACL*, pages 641–651, 2018.

[Williams, 1992] Ronald J. Williams. Simple statistical gradient following algorithms for connectionist reinforcement learning. *Machine Learning*, pages 8:229–256, 1992.

[Xu *et al.*, 2016] Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. Question answering on freebase via relation extraction and textual evidence. In *Proceedings of ACL*, pages 2326–2336, 2016.

[Yang and Chang, 2015] Yi Yang and Ming-Wei Chang. S-mart: Novel tree-based structured learning algorithms applied to tweet entity linking. In *Proceedings of ACL-IJCNLP*, pages 504–513, 2015.

[Yao and Durme, 2014] Xuchen Yao and Benjamin Van Durme. Information extraction over structured data: Question answering with freebase. In *Proceedings of ACL*, pages 956–966, 2014.

[Yih *et al.*, 2015] Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of ACL-IJCNLP*, pages 1321–1331, 2015.

[Yih *et al.*, 2016] Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of ACL*, pages 201–206, 2016.

[Yu *et al.*, 2017] Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cicero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. Improved neural relation detection for knowledge base question answering. In *Proceedings of ACL*, pages 571–581, 2017.

[Zhang *et al.*, 2017] Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J. Smola, and Le Song. Variational reasoning for question answering with knowledge graph. In *Proceedings of AAAI*, pages 6069–6076, 2017.