Question Answering over Freebase via Attentive RNN with Similarity Matrix based CNN

Yingqi Qu[1] , Jie Liu[1], Liangyi Kang[1] , Qinfeng Shi[2], and Dan Ye[1]

[1] Institute of Software, Chinese Academy of Sciences, Beijing, China
[2] School of Computer Science, The University of Adelaide, Australia
{quyingqi16, ljie, kangliangyi15, yedan}@otcaix.iscas.ac.cn
javen.shi@adelaide.edu.au

Abstract. With the rapid growth of knowledge bases (KBs), question answering over knowledge base, a.k.a. KBQA has drawn huge attention in recent years. Most of the existing KBQA methods follow so called encoder-compare framework. They map the question and the KB facts to a common embedding space, in which the similarity between the question vector and the fact vectors can be conveniently computed. This, however, inevitably loses original words interaction information. To preserve more original information, we propose an **a**ttentive **r**ecurrent neural network with **s**imilarity **m**atrix based **c**onvolutional **n**eural **n**etwork (AR-SMCNN) model, which is able to capture comprehensive hierarchical information utilizing the advantages of both RNN and CNN. We use RNN to capture semantic-level correlation by its sequential modeling nature, and use an attention mechanism to keep track of the entities and relations simultaneously. Meanwhile, we use a similarity matrix based CNN with two-directions pooling to extract literal-level words interaction matching utilizing CNN's strength of modeling spatial correlation among data. Moreover, we have developed a new heuristic extension method for entity detection, which significantly decreases the effect of noise. Our method has outperformed the state-of-the-arts on SimpleQuestion benchmark in both accuracy and efficiency.

Keywords: Question Answering, Knowledge Base, Deep Learning, Freebase.

1 Introduction

In recent years, several large-scale general-purpose knowledge bases have emerged, including YAGO [8], Freebase [9], NELL [10] and DBpedia [11], and people are seeking effective ways to access the rich knowledge in them. There are several languages designed for querying the KBs, however they are not very user friendly --- users need to be familiar with the language grammars and the vocabulary of the KBs. Alternatively, question answering over knowledge base (KBQA) allows users to directly use natural language to query the KBs, and has attracted much attention

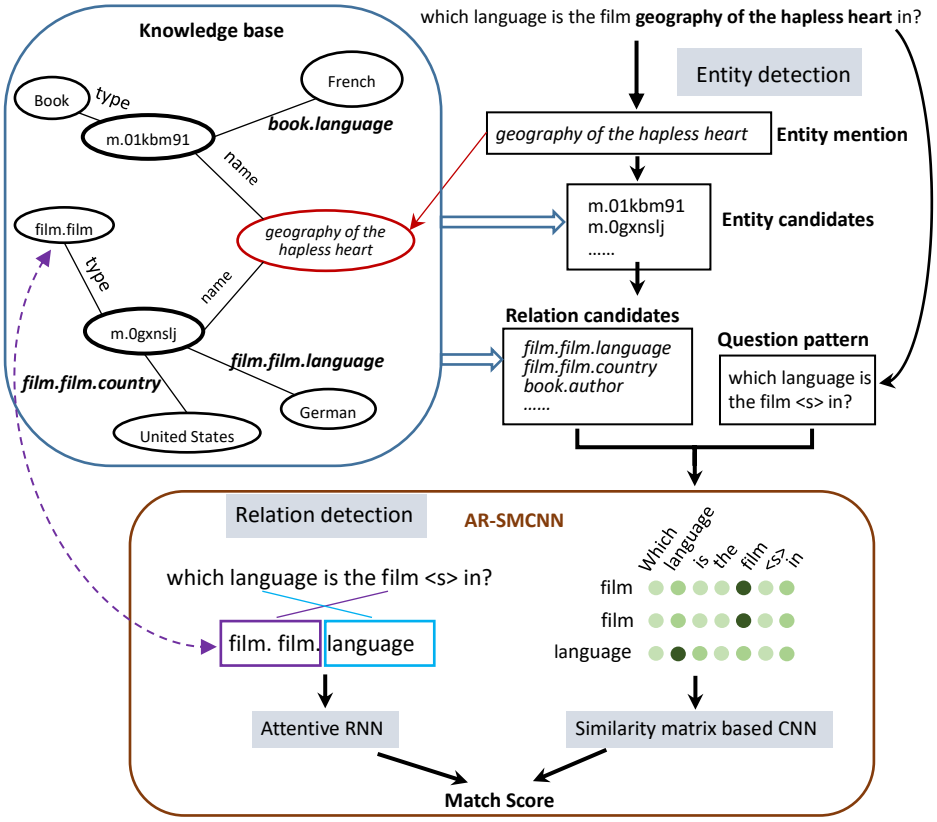

Fig. 1. Two-step process to answer single-relation questions. Given a query question, we use Entity Detection (Right) which produces an *entity mention* that is used to search Freebase (Left). All the entities whose name or alias matches the entity mention are considered as *entity candidates*, and all of the relations connected to the entity candidates are *relations candidates*. The original question is converted into *question pattern* by replacing the entity mention with symbol <s>. Our AR-SMCNN model (Bottom), which we defer the details to Section 3.3, detects the correlation between question pattern and the relation candidates.

recently. Though KBQA has brought convenience to the end-users, it is challenging due to the discrepancy of the structured KBs and the unstructured natural language questions. In this paper, we mainly focus on the questions that can be answered with a single fact in the knowledge base, which are the most common types in KBQA [23, 25]. Answering single-relation questions is the foundation of handling other more complex and multiple-relations questions, and this kind of task is still far from being solved because there are a large number of paraphrases of the same question making the task of mapping a question to a particular triple in KB difficult.

Our work is mainly based on Freebase [9], where triples are in the form of subject-relation-object. We tackle the question answering task in two steps: (1) extract the topic of the question, which is corresponds to the subject in triples; (2) predicting the relation that best describes the subject and the answer. As long as the predictions of

subject and relation are both correct, the answer is certain. The process is illustrated in Figure 1.

Most of the existing deep learning methods for KBQA follow an encoder-compare framework (i.e. [1, 4-6, 15]), where firstly use n-gram of question text to search Freebase to generate a set of candidate KB facts, and then map the question and KB elements (entity name and relation name) to a common embedding space via a deep neural network. The correlation between question and KB facts is calculated by the similarity between two embedding vectors. These methods have three main deficiencies: (1) encoded embedding vectors only contain semantic information, thus the calculation of similarity between them can only catch a high level correlation and loses original words interaction information; (2) the set of entity candidates is formed by all the entities whose name is a substring of the question, which introduces a lot of noise; (3) the encoding of entity name in [4,5] is on character level, thus the entity matching model only detects text surface matching, which cannot distinguish the entities with the same name.

To deal with these problems, we propose an **a**ttentive **r**ecurrent neural network with **s**imilarity **m**atrix based **c**onvolutional **n**eural **n**etwork (AR-SMCNN) model and a new entity detection method. AR-SMCNN leverages the complementary advantages of both convolutional neural network (CNN) and recurrent neural network (RNN) to capture comprehensive hierarchical matching information, where CNN is good at extracting position in variant features and dealing with spatially related data, and RNN is good at modeling units in sequence and dealing with temporal signals (see Yin et al. [22]).

The bottom of Figure1 shows two different inputs of our AR-SMCNN model. The left part is attentive RNN, which catches semantic-level correlation by a standard encoder-compare structure that encode question and relations separately. Here we divide relations of Freebase into two parts, represent the subject type and the genuine relationship respectively (based on the observation that the first two part of KB relation "*film.film*" is identical to subject type). Attention mechanism can catch the different contribution of each words in the question respect to these two parts. Meanwhile, the focus on the subject type can distinguish entities with the same name, which solves the leftover problem from entity detection implicitly. As shown in Figure 1, entity candidate *m.01kbm91* and *m.0gxnslj* have same name, but different types. We can exclude the entities whose type differ from "*film*", which the question is asking about. Furthermore, such a practice also helps to alleviate the OOV problem because it reduces the number of relation categories by around a half.

The right part is similarity matrix based CNN, which detects literal-level matching between question and relations words. Inspired by Pang et al. [12], we construct a similarity matrix whose entries represent the similarity between question words and relation words. Based on it, we utilize CNN with two-direction max-pooling to extract words interaction matching from perspective of question and relation respectively. This is based on the observation that KB relations often corresponds to several keywords or rephrased tokens in the question. Similarity matrix formulates these corresponding relationships as an interaction structure, and a convolutional layer over it captures abundance matching patterns.

In addition, we also propose an entity detection approach. We come up with a heuristic extension method to get entity candidates and omit the entity matching neural network. This simple and efficient method achieves same effect without additional training model, and also eliminates the noise introduced by some irrelevant entities that have common substrings with question accidentally.

Our system achieves state-of-the-art result on SimpleQuestion dataset [3]. We also compare with previous works on entity detection task and relation detection task separately, and both of them have a competitive effect. To summarize, the main contributions of this paper is that we propose a novel approach to deal with single-relation KBQA, which includes (1) an attentive RNN with similarity matrix based CNN (AR-SMCNN) model for relation detection, which captures semantic-level correlation along with literal-level matching information, (2) and a heuristic extension method for entity detection, which is efficient and effective.

2 Related Work

Single-relation question answering over KB was first investigated in [23] through PARALEX dataset against knowledge base Reverb [24]. An important line of research in this domain investigates semantic parsing approaches to translate questions into structured KB queries [19-21], which requires complex NLP pipelines.

Another line of research tackles the problem by deep learning powered similarity matching. The core idea is to learn semantic representations of both the question and the KB elements with an encoder-compare framework, such that the correct supporting evidence will be the nearest neighbor of the question in the learned vector space [6]. The main difference among these approaches lies in the encoder model (RNN or CNN) and the input granularity (word level or character level). Dai et al. [6] investigate a relation-level RNN-based approach with a reasonable probability interpretation. Yin et al. [4] split relations to word sequences, employ CNNs and propose an attentive max-pooling approach to improve the model. Lukovnikov et al. [1] also use word-level relation representations, but the network they use is BiGRU. To make a finer division on relation, Golub and He [5] propose a character-level approach based on the attention-enhanced architecture, where attention was introduced to better handle longer sequences. Yih et al. [14] use character tri-grams as inputs on both question and relation sides. Yu et al. [15] concentrate on two granularities of relation: word-level and relation-level. They propose a hierarchical RNN enhanced by residual learning which detects different levels of abstraction.

Our approach is closely related to the second line of research, but we construct an additional CNN that does not follow the traditional encoder-compare framework.

The application of CNN in natural language processing (NLP) has achieved great progress recently. [29, 30] first applied CNN in NLP field, getting competitive results in sentence classification tasks, sentiment prediction tasks and so on. Yin et al. [22] compared RNN and CNN and gave the conclusion that CNN is good at extracting position in variant features and dealing with spatially related data. Wang et al. [32] integrated the merits on extracting different aspects of linguistic information from

both RNN and CNN for modeling text. Pang et al. [12] proposed a text matching method that references the success of CNN in image recognition. The model can automatically capture important matching patterns such as unigram, n-gram and n-term at different levels. Inspired by them, we construct a similarity matrix of question and relation, and utilize CNN to catch original words interaction matching. Our entire model takes advantage of the complementary strengths of RNN and CNN, and is able to capture comprehensive correlation between question and relation.

For entity detection, previous works use n-gram of question words to search for entities whose name has common substring with question, like [5, 1, 3]. Then they use a neural network to catch surface matching between question and these entity names, along with other syntactic indicators (such as edit distance, LCCS (longest consecutive common subsequence)). Golub and He [5] use character-level LSTM to encode question and entity names respectively, while Yin et al. [4] use character-level CNN to encode both of them; Lukovnikov et al. [1] encode the question with both word-level and character-level. Besides, in order to narrow down the search range, Yin et al. [4] and Dai et al. [6] use an additional BiLSTM-CRF tagging model to detect entity text span in advance. Although this makes some improvement, it still takes a long time to access the knowledge base during the searching process and introduces a lot of noise. Unlike them, we come up with a heuristic extension method to obtain entity candidates, no need to build an additional neural network, which can reduce complexity and increase efficiency significantly. It also eliminates the noise introduced by some irrelevant entities that have common substrings with question.

In order to distinguish entities with the same name, Dai et al. [6] and Lukovnikov et al. [1] use entity types as additional information. [6] regard it as a multi-classification problem and train a model to classify questions into entity types, while [1] add this information into matching model and encode the entities with the concatenate of character-level entity names and word-level types. Other prior works ignore this problem and do nothing in their model. We observed that the definition of relation in Freebase implicitly contains entity type information, so we leave this work to relation detection and use an attention mechanism to focus on this part.

3 Approach

In the following we describe our approach for KBQA which consists of two components: (1) an entity detection method for finding topic entities and (2) a relation detection model for capturing correlation between question and candidate relations.

3.1 Task Definition

We first describe the knowledge base our work based on. Freebase is a structured knowledge base in which entities are connected by predefined relations. A triple (*subject, relation, object*), denoted as (*s, r, o*), describes a fact.

The entire process is illustrated in Figure1. We assume that single-relation questions can be answered by querying the knowledge base with a single subject and relation

argument. Hence, only the tuple (*s*, *r*) is needed to match the question. As long as the predictions of *s* and *r* are both correct, we can get the answer directly (which is obviously corresponding to *o*). According to the above assumptions, the problem can be solved by the following two steps:

(1) Identify candidate entities in Freebase that the question refers to. Given a question *Q*, we need to find out the entity mention *X*, then all entities whose name or alias is identical with entity mention will make up entity candidates *E*. Now all the entities in *E* have the same entity name, as we cannot differentiate them for now.

(2) All the relations connected to the entities in *E* are regarded as candidate relations, named as *R*. We convert the question into pattern *P* which is created by replacing the mention in the question with <e>. To find out the relation that is truly relevant to the question, we compare *P* with each relation in *R* and score them, then the relation with the highest score can be regarded as the final result.

3.2 Entity Detection

The entity detection process is illustrated in Figure 2. Given a question *Q*, we train a bidirectional LSTM network [26] like a sequential label task, which can be regarded as a binary classification problem that predicts whether each word in a sentence belongs to entity mention. Then, we get a set of words with a positive label, which is denoted as *C*. These words may not be consecutive, and then we use following heuristic method to get entity mention *X*.

(1) Combine the adjacent words (ignore the gap <= 1) in *C* to form a substring *S*. If there is more than one substring, just keep the longest one.

(2) Find out all entities in Freebase whose name or alias exactly equals to *S*. These entities form entity candidates *E*, and *S* is regarded as entity mention. If there's nothing matches S, continue to step (3).

(3) Based on the fact that the words in *S* are most likely to constitute entity mention *X*, thus there is a huge probability that *X* can be generated based on *S*. So we take *S* as the center and keep finding *X* around it. Specifically, we expand or narrow at most 2 words around *S* to get S',[1] and use S' to find corresponding entities. Once a match is found, *E* and *X* are determined.

(4) If there is still not any match found, then we use each word in *S* to search for entities whose name contain the word. We keep the entities who have the longest common subsequence with question as the entity candidates *E*, and the common subsequence will be *X*. This step is similar to previous methods, but the probability of this happening here is less than 0.2%.

The purpose of (1) is to combine the decentralized words, which follows the presumption that if there is a word predicted to be negative label, but its neighbors are positive, it must be predicted wrong. On the basis of this, the following operations will improve the recall. The subjects in entity candidates are sorted by their out-degrees, which is the total number of relations they connected to.

---

[1] The process is [*l+1, r+1, l-1, r-1, l+2, r + 2,* ⋯], where *l+1* means add one word from left side, *r-1* means discard one word from right side.

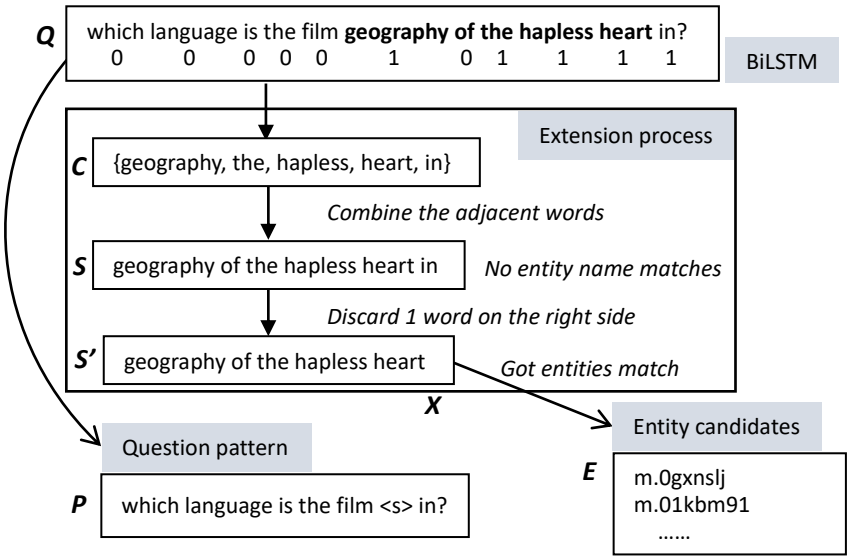

Fig. 2. The entity detection process. We train a BiLSTM network to get a set of words who are possible to be a part of entity mention; after the extension process, we get a set of entity candidates, and generate the question pattern based on it.

3.3 Relation Detection

Given a question pattern *P*, for each relation r^k in the pool of relation candidates *R*, we compute a matching score $S(P, r^k)$ that represents the correlation between them. The final prediction is $\hat{r}^k$ with

$$\hat{r}^k = \underset{r^k \in R}{argmax}(S(P, r^k)). \tag{1}$$

We consider two granularities of correlation between *P* and r^k by our AR-SMCNN model, which are the semantic-level and literal-level matching. In the following sections, we will introduce two components of the model respectively, with the theories and implementations. Figure 3 shows the holistic structure.

Semantic-level. To catch semantic-level matching between question and relations, we construct an attentive RNN as an encoder-compare framework. We find out that the relations in Freebase contains two aspects of information, one is identical to subject's type, while the other describes the relationship between subject and object. i.e., the question pattern "*which language is the film <e> in*" corresponds to the relation < *film.film.language* >. The first two parts of relation "*film.film*" represent the type of subject, and the last part "*language*" represents the genuine relationship between subject and the answer "*German"*. These two parts correspond to different words of question, thus each word in the question should give different contribution to question encoding. Based on this observation, we encode two parts of relation separately, and use an attention mechanism on question encoder to match both of them.

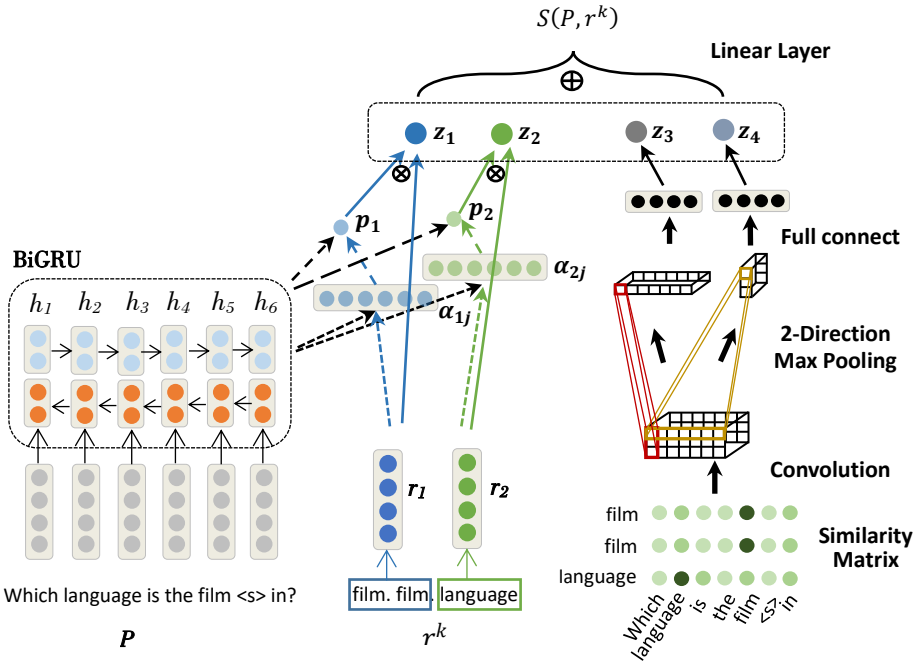

Fig. 3. The proposed AR-SMCNN model. The left part is attentive BiGRU, and the right part is CNN on similarity matrix. The features z_1, z_2, z_3, z_4 is concatenated together and pass through a linear layer to get final score $S(P, r^k)$.

Relation encoding. For each relation in *R*, we separate them into two parts as we described above, and transform each part to its trainable embedding (which is randomly initialized) to get their vector representations r_1 and r_2.

Question pattern encoding. Each word in the question is transformed to its word embedding $\{q_1, \cdots, q_L\}$, then the embeddings are fed into a bidirectional GRUs [27] network to get hidden representations $\mathrm{H}_{1:L} = [h_1; \cdots; h_L]$ (each vector h_i is the concatenation between forward/backward representations at time *i*). Each part of the relation pays different attention to the question and decides how the question is represented. The extent of the attention is used as the weight of each word in the question. Thus, for relation representation r_i, its corresponding question pattern representation p_i is calculated as:

$$p_i = \sum_{j=1}^{L} \alpha_{ij} h_j \,, \tag{2}$$

$$\alpha_{ij} = \frac{\exp(w_{ij})}{\sum_{k=1}^{L} \exp(w_{ik})} \,, \tag{3}$$

$$w_{ij} = v^T tanh\left(W^T \left[h_j; r_i\right] + b\right) \,, \tag{4}$$

where $i \in \{1, 2\}$, α_{ij} denotes the attention weight of the *j*-th word in the question in

terms of relation aspect r_i. L is the length of the question. Let the dimension of r_i be m, the dimension of h_i be n, then W and v are the parameters to be learned with $W \in R^{c\times(m+n)}$, $v \in R^{1\times c}$ (where c is a hyperparameter).

Similarity measure. Now we have the representation of question pattern and relation, their similarity is calculated by the following equation

$$z_i = p_i \otimes r_i \quad (i = 1,2). \tag{5}$$

The operation $\otimes$ here is the dot product of two vectors. We have also tried other methods to calculate the interaction between them, such as cosine, bilinear, Manhattan distance and GESD [2], but they did not result in improvements and was slightly slower during training. It shows that simple dot product is enough in this task.

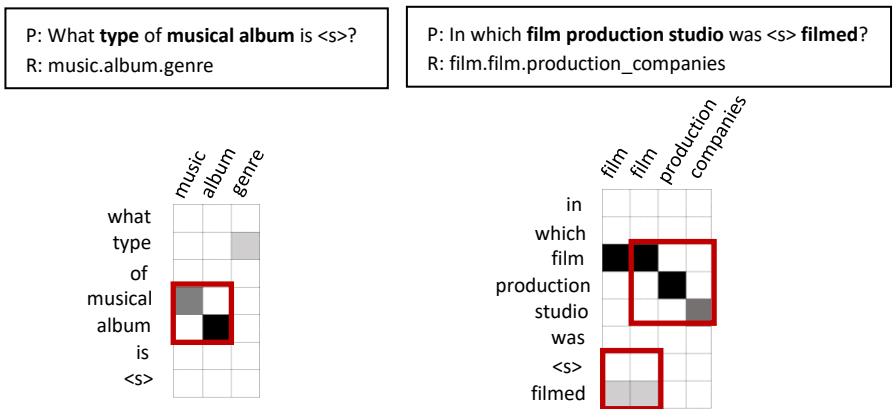

Fig. 4. Two examples of similarity matrix. The degree of darkness represents the similarity between two words. We can see that there are different kinds of matching patterns (labeled in red squares), and convolutional kernels can extract these features.

Literal-level. When considering the correlation over literal-level, we treat it as a text matching problem. We find out that some words (or phrases) in question and relation have different expressions (or orders), even if they express the same meaning. Like the examples shown in Figure 4, the pairs of words (*musical*, *music*) (*type*, *genre*) (*studio*, *companies*) all indicate to the similar topic. An encoder-compare model cannot catch these words interaction information, because the representation after encoder only reserves high-level semantic information. Inspired by Pang et al. [12], we construct a similarity matrix whose entries represent the similarities between question words and relation words, regarding it as a two-dimensional image to utilize convolutional layer to catch the matching features. The following describes details.

Similarity matrix. We construct a similarity matrix M, where each element M_{ij} standing for the basic interaction, i.e. similarity between u_i and v_j (u_i and v_j denotes the i-th and the j-th word embedding in question and relation respectively):

$$M_{ij} = u_i \otimes v_j \,, \tag{6}$$

where $\otimes$ stands for a general operator to obtain the similarity, which is cosine here. Unlike strict textual matching, this matrix can catch the similarity of words with different forms or expressions.

Convolution layer. A typical convolutional kernel can extract different levels of matching patterns, such as word level or phrase level matching (even in different order). More specifically, the *k*-th kernel w^k scans over the whole similarity matrix *M* to generate a feature map g^k:

$$g_{i,j}^k = \sigma(\sum_{s=0}^{r_k-1}\sum_{t=0}^{r_k-1} w_{s,t}^k \cdot M_{i+s,j+t} + b^k), \quad (7)$$

where r_k denotes the size of the *k*-th kernel. Here we use square kernels, and ReLU [16] is adopted as the active function σ.

Max pooling layer from 2-directions. We use two different pooling kernels on the top of feature map g^k. The sizes are $1 \times d_1$ and $d_2 \times 1$ respectively, where d_1 and d_2 denote to the width and length of similarity matrix:

$$y_i^{(1,k)} = \max_{0 \le t < d_1} g_{i,t}^k, \quad (9)$$

$$y_j^{(2,k)} = \max_{0 \le t < d_2} g_{t,j}^k. \quad (10)$$

The key idea is that it retains the max matching feature from the perspective of both question and relation, which is for each word in the question, the max matching score among every relation words, and for each word in the relation, the max matching score among every question words. This is better than using a square pooling kernel, because we place more emphasis on the max matching score of a single word in this task.

Fully connected layer. A MLP (Multi-Layer Perception) is used to produce the final feature. Taking two-layer perceptron for each pooling result, we will obtain:

$$z_3 = W_2\sigma\big(W_1[y^{(1,0)}; y^{(1,K)}] + b_1\big) + b_2, \quad (11)$$

$$z_4 = W_2\sigma\big(W_1[y^{(2,0)}; y^{(2,K)}] + b_1\big) + b_2, \quad (12)$$

where *K* denotes the total number of kernels, and $[y^{(i,0)}; y^{(i,K)}]$ is the concatenation of *K* pooling layer outputs, W_i is the weight of the *i*-th MLP layer and σ denotes the activation function, which is ReLU here.

Combination. After two granularities of matching, we get four features (z_1, z_2, z_3, z_4) referring to different aspects, where z_1 and z_2 denote to semantic relevance with subject type and relationship respectively, z_3 and z_4 denote to literal matching from both question and relation sides. We utilize a linear layer to learn their respective contribution for holistic matching score:

$$S(P, r^k) = Sigmoid(W^T[z_1; z_2; z_3; z_4] + b). \quad (13)$$

The model described above is trained with a ranking loss to maximizing the margin between the gold relation r^+ and other relations r^- in the candidate pool R:

$$loss(P, r^+, r^-) = \sum_{(P,r^+)\in\mathcal{D}} \max\left(0, \gamma + S(P, r^-) - S(P, r^+)\right), \quad (14)$$

where γ is a constant parameter.

4 Experiments

4.1 Dataset

We verify our proposed approach on SimpleQuestion dataset. SimpleQuestion benchmark, released by Bordes et al. [3], provides a set of single-relation questions; each question is accompanied by a ground truth fact, which is a triple in Freebase. The dataset is split into train (79,590), valid (10,845) and test (21,687) sets. This benchmark also provides two subsets of Freebase: FB2M and FB5M. The former includes 2M entities, while the latter contains 5M entities.

4.2 Experiment Setting

In entity detection process, we generate training set by mapping the gold subject back to the text to label the text span for each question. More specifically, for each (question, fact) pair, we match the names (or aliases) of subject with question, and take the longest one as its gold entity text span.

During training, all word embeddings are initialized using the pretrained GloVe [13] with 300 dimensions. The embeddings of relations are randomly initialized with 150 dimensions. The hidden layer of BiGRU has size 200, and hinge loss margin γ is set to 1. The number of CNN channel is 8 and the kernel size is set to be 3×3. For optimization, parameters are trained using Adam [31] with a learning rate of 0.0005 in a mini-batch setting with batch size 64. Dropout is used to regularize BiGRU and CNN in our experiment and is set to 0.3.[2]

Negative sampling sizes is 50, which is much smaller than other approach (Dai et al. [6] use 1024). We have tried other larger negative sampling sizes, such as 75 and 100, but there were no obvious benefits. This indicates the conciseness and effectivity of our model.

5 Results

In this section, we will show the overall results on SimpleQustion dataset, where we train and predict following the proposed method. In addition, we analyze the result of entity detection and relation detection separately to show their effectiveness. We also visualize the attention distribution over question words in the encoding process via

[2] For more details, source code is available at https://github.com/quyingqi/kbqa-ar-smcnn

different relation aspects, to show how the attention mechanism works.

5.1 Overall results

We compare our results with six recent state-of-the-art works that developed QA models on the SimpleQuestion dataset. These works include the Memory Network approach of Bordes et al. [3], the character-level attention-enhanced encoder-decoder approach by Golub and He [5], the end-to-end multi-level GRU based approach by Lukovnikov et al. [1], the word-level RNN-based approach by Dai et al. [6], the attentive CNN based approach by Yin et al. [4], and the hierarchical residual LSTM based approach by Yu et al. [15].

Table 1. Overall accuracy on SimpleQuestion test sets.

Top K	FB2M	FB5M
Memory Network (Bordes et al., 2015)	62.7	62.2
Char-level(Golub and He, 2016)	70.9	70.3
End-to-end (Lukovnikov et al., 2017)	71.2	-
CFO (Dai et al., 2016)	-	75.7
AMPCNN (Yin et al., 2016)	76.4	-
HR-BiLSTM (Yu et al., 2017)	77.0	-
AR-SMCNN (our Method)	**77.9**	**76.8**

We follow Bordes et al. [3] in comparing the predicted entity-relation pair to the ground truth. A question is counted as correct if and only if the entity we select and the relation we predict match the ground truth. As shown in Table 1, our approach achieves accuracies of 77.9% and 76.8% on the FB2M and FB5M settings, outperforming the previous state-of-art results by 0.9% and 1.1%, respectively.

Furthermore, our approach is not only effective, but also efficient. All experiments are carried out on a machine with an Nvidia GTX1080 GPU. Our training process cost around one hour with 20 epochs and achieves best result.

5.2 Entity Detection

We compare our entity detection result with several baselines, including focused pruning method by Dai et al. [6], active entity linking method by Yin et al. [4], and candidate generation method by Lukovnikov et al. [1]. Table 2 shows the recall of top K entity candidates ($K \in \{1, 5, 10, 20, 50, 100, 400\}$), where recall is computed as the percentage of questions for which the top K of entity candidates includes the right subject entity. The first row can be viewed as the accuracy of entity detection. Table 3 shows the overall recall, which is the percentage of questions for which the entity candidate set contains the expected entity.

As we explained in section 3.2, our extension method detects the entity mention in the question straightway, omit the entity matching process as [1, 3, 6] did. As a result, it will reduce the recall to some extent. But on the other hand, our generated

Table 2. The recall of top K entity candidates.

Top K	Yin et al.	Lukovnikov et al.	Our method
1	73.6	71.2	74.5
5	85.0	85.4	86.0
10	87.4	88.4	88.5
20	88.8	-	90.2
50	90.4	-	91.9
100	91.6	-	93.1
400	-	93.7	95.1

Table 3. The overall recall of entity candidates and the average size of candidates. The last line shows that removing the extension process causes a big drop in performance.

Method	Recall	Ave. size.
Dai et al.	94.9	
Yin et al.	96.7	162
Our method	95.8	57
w/o extension	93.8	

candidates only contain the entities with the same name, which greatly reduce the scale of candidates, and avoids noise produced by some irrelevant entities who have common substring with question accidentally.

Table 3 shows that we get 95.8% overall recall, which is slightly lower than state-of-the-art result 96.7% from Yin et al. [4], but our average size of candidate set is one third of theirs[3]. Meanwhile, Table 2 shows that from top 1 to top 400, our recall is higher than both [4] and [1]. It proves that we have more useful information in a smaller candidate set, and the correct entity has a higher ranking.

5.3 Relation Detection

We test our proposed relation detection model on the dataset released by Yin et al. [4]. They have already replaced the entity mention by a special symbol <e> and constructed relation candidates, so we have the same starting point to make the comparison of this subtask meaningful[4].

Table 4 shows the results on relation detection subtask. The baselines include BiCNN [14], AMPCNN [4] and HR-BiLSTM [15]. All of them follow the encoder-compare framework, which firstly map the question and relation as vectors and then get the semantic similarity by vector comparison. It shows that our approach outperforms state-of-the-art result by Yu et al. [15], which uses a hierarchical residual BiLSTM model to encode the question and relation on both word level and relation name level. Our AR-SMCNN model benefits from the attention of two relation aspects and the words interaction matching, which differ with previous methods.

Ablation experiments are further carried out to analyze the impacts of each component of our model. The lower part of Table 4 shows the ablation results. The BiLSTM results (on the fourth and fifth line) are released by Yu et at. [15]. "rel_name" denotes relation names, and "rel_separated" represents that relations are divided into two parts. "SimMatrix-CNN" denotes similarity matrix based CNN model. Based on the experimental results, we have following observations.

1) Taking relation name as an entirety, BiGRU performs a little bit better than

[3] This is calculated based on the entity linking result they released at https://github.com/Gorov/SimpleQuestions-EntityLinking

[4] They released their dataset at https://github.com/Gorov/KBQA_RE_data

Table 4. Accuracy on relation detection tasks.

Model	Relation Views	Accuracy
BiCNN (Yih et al., 2015)	char-3-gram	90.0
AMPCNN (Yin et al., 2016)	words	91.3
HR-BiLSTM (Yu et al., 2017)	words + rel_name	93.3
BiLSTM w/words	words	91.2
BiLSTM w/rel_name	rel_name	88.9
BiGRU w/rel_name	rel_name	90.3
BiGRU w/rel_separated	rel_separated	92.0
Attentive-BiGRU	rel_separated	92.8
SimMatrix-CNN	words	91.2
SimMatrix-CNN +2dir-pooling	words	91.8
AR-SMCNN	words + rel_separated	**93.7**

BiLSTM (90.3% and 88.9% respectively). On this basis, splitting relation into two parts and encoding them separately results in a significant improvement (92.0%). This demonstrates that two aspects of the entire relation can confuse its semantic representation, making it difficult to reserve the overall information in one vector.

2) Adopting attention mechanism on BiGRU can also give a performance boost (0.8%). It indicate that giving different question representations respect to two aspects of relation can facilitate the capture of matching information.

3) The performance of a single similarity matrix base CNN is almost the same as AMPCNN and word-level BiLSTM. When two-directions pooling employed, accuracy increases from 91.2% to 91.8%, which proves the effectiveness of two max-pooling kernel.

4) Combining these components together causes a big improvement, achieves the best accuracy with 93.7%, which demonstrate that the attentive RNN and similarity matrix base CNN have complementary strength, and our final model can successfully capture comprehensive hierarchical correlation between question and relation.

5.4 Effectiveness of attention

According to the results in section 5.3, the attention mechanism plays a critical role in achieving the best performance. To gain a deeper understanding of its effectiveness, we visualize the attention distribution of the question words in question encoding process (see Figure 5).

We can find out that for two relation aspects, each question word learns a different weight and capture the attention properly. i.e., in the first example, one aspect of relation "*music.genre"* pays great attention to words "*musical*", "*genre*" and "*classified*", while another aspect *artist* mainly focuses on word "*artist*". It shows that AR-SMCNN learns a reasonable alignment between question words and relation aspects.

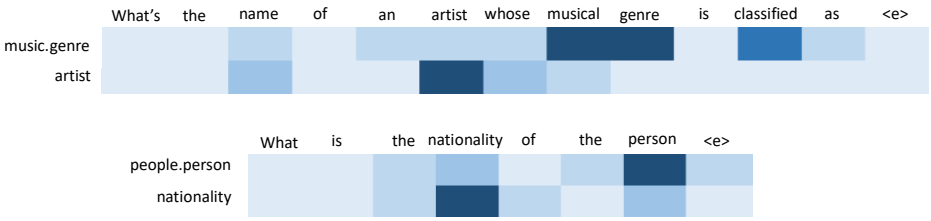

Fig. 5. The visualized attention heat map. Depth of the color represents the attention weights, and the darker the higher.

6 Conclusions

In this article, we presented a novel, neural network based approach for answering single-relation questions over largescale knowledge base, which leverage complementary strength of RNN and CNN to capture semantic and literal relevance information. While keeping the model simple by omitting the entity matching model, the proposed approach achieves competitive results. Even though the presented approach is restricted to single-relation questions, this work can serve as a foundation for the future development of more advanced neural QA approaches that can handle more complex questions. In future work, we will extend our approach to deal with multi-relation problems, with an additional part of constraint detection. We will also investigate new emerging datasets like GraphQuestions [17], ComplexQuestions [18] and Lc-quad [28] to handle more characteristics of general QA.

References

1. Lukovnikov, D., Fischer, A., Lehmann, J., et al.: Neural network-based question answering over knowledge graphs on word and character level. In: 26th www, pp. 1211–1220. (2017).
2. Feng, M., Xiang, B., Glass, M. R., et al.: Applying deep learning to answer selection: A study and an open task. In: ASRU. pp. 813-820. IEEE (2016).
3. Bordes, A., Usunier, N., Chopra, S., et al.: Large-scale simple question answering with memory networks. arXiv preprint arXiv:1506.02075 (2015).
4. Yin, W., Yu, M., Xiang, B., et al.: Simple question answering by attentive convolutional neural network. arXiv preprint arXiv:1606.03391 (2016).
5. Golub, D., He, X.: Character-level question answering with attention. arXiv preprint arXiv:1604.00727 (2016).
6. Dai, Z., Li, L., Xu, W.: CFO: Conditional focused neural question answering with large-scale knowledge bases. arXiv preprint arXiv:1606.01994 (2016).
7. Yang, H., Yang, H., Yang, H.: A hybrid framework for text modeling with convolutional RNN. In: ACM SIGKDD, pp. 2061-2069. ACM (2017).
8. Suchanek, F.M., Kasneci, G., Weikum, G.: YAGO: A core of semantic knowledge unifying Wordnet and Wikipedia. In: 16th WWW, pp. 697–706. ACM (2007).
9. Bollacker, K., Evans, C., Paritosh, P., et al.: Freebase: a collaboratively created graph database for structuring human knowledge. In: ACM SIGMOD, pp. 1247–1250. ACM (2008).

10. Carlson, A., Betteridge, J., Kisiel, B., et al.: Toward an architecture for never-ending language learning. In: 24th AAAI, vol.42, pp. 1306-1313. AAAI Press (2010).
11. Lehmann, J.: Dbpedia: a large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web 6(2), 167-195 (2015).
12. Pang, L., Lan, Y., Guo, J., et al.: Text matching as image recognition. In: 30th AAAI Conference on Artificial Intelligence, pp. 2793–2799. (2016).
13. Jeffrey Pennington, Richard Socher, Christopher D.: Manning. Glove: Global vectors for word representation. In: Proceedings of the 2014 EMNLP, pp. 1532–1543. (2014).
14. Yih, S.W.t., Chang, M.W., et al.: Semantic parsing via staged query graph generation: Question answering with knowledge base. In: 53rd ACL, vol. 1, pp. 1321–1331. (2015).
15. Mo Yu.: Improved neural relation detection for knowledge base question answering. arXiv preprint arXiv:1704.06194. (2017).
16. Dahl, G. E., Sainath, T. N., Hinton, G. E.: Improving deep neural networks for lvcsr using rectified linear units and dropout. In: IEEE ICASSP, vol. 26, pp. 8609–8613. IEEE (2013)
17. Su, Y., Sun, H., Sadler, B., et al.: On Generating Characteristic-rich Question Sets for QA Evaluation. In: EMNLP, pp. 562-572. (2016)
18. Bao, J., Duan, N., Yan, Z., et al.: Constraint-based question answering with knowledge graph. In: 26th COLING, pp. 2503–2514. (2016)
19. Berant, J., Liang, P.: Semantic parsing via paraphrasing. In: 52nd Annual Meeting of ACL, pp. 1415–1425. (2014).
20. Reddy, S., Lapata, M., Steedman, M.: Large-scale semantic parsing without question-answer pairs. In: Transactions of the ACL, 2:377–392 (2014).
21. Berant, J., Chou, A., Frostig, R., Liang, P.: Semantic parsing on freebase from question-answer pairs. In: EMNLP, vol. 2, pp. 6. (2013)
22. Yin, W., Kann, K., Yu, M., Schütze, H.: Comparative study of cnn and rnn for natural language processing. arXiv preprint arXiv:1702.01923. (2017).
23. Fader, A., Zettlemoyer, L., Etzioni, O.: Paraphrase-driven learning for open question answering. In: Proceedings of ACL, pp. 1608–1618. (2013).
24. Fader, A., Soderland, S., Etzioni, O.: Identifying relations for open information extraction. In: Proceedings of EMNLP, vol.17, pp. 1535–1545. (2011).
25. Yih, W. T., He, X., Meek, C.: Semantic parsing for single-relation question answering. In: Proceedings of ACL, pp. 643-648. (2014).
26. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation, 9(8), 1735–1780 (1997).
27. Cho, K., Van Merrienboer, B., Gulcehre, C., et al.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. Computer Science. (2014).
28. P. Trivedi, G. Maheshwari, M. Dubey, J. Lehmann. Lc-quad: A corpus for complex question answering over knowledge graphs. In: Proceedings of ISWC, pp. 210–218. Springer (2017).
29. Kalchbrenner, N., Grefenstette, E., Blunsom, P.: A convolutional neural network for modelling sentences. arXiv preprint arXiv:1404.2188 (2014).
30. Kim, Y.: Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882 (2014).
31. Kingma, D., Ba, J. Adam: a method for stochastic optimization. Computer Science. (2014).
32. Wang, C., Jiang, F., Yang, H.: A Hybrid Framework for Text Modeling with Convolutional RNN. In: ACM SIGKDD, pp. 2061-2069. ACM. (2017).