How Question Generation Can Help Question Answering over Knowledge Base

Sen Hu, Lei Zou(✉), and Zhanxing Zhu

Peking University, Haidian Qu, China
{husen,zoulei,zhanxing.zhu}@pku.edu.cn

Abstract. We study how to improve the performance of Question Answering over Knowledge Base (KBQA) by utilizing the factoid Question Generation (QG) in this paper. The task of question generation (QG) is to generate a corresponding natural language question given the input answer, while question answering (QA) is a reverse task to find a proper answer given the question. For the KBQA task, the answer could be regarded as a fact containing a predicate and two entities from the knowledge base. Training an effective KBQA system needs a lot of labeled data which are hard to acquire. And a trained KBQA system still performs poor when answering the questions corresponding with unseen predicates in the training process. To solve these challenges, we propose a unified framework to combine the QG and QA with the help of knowledge base and text corpus. The models of QA and QG are first trained jointly on the gold dataset, then the QA model is fine tuned by utilizing a supplemental dataset constructed by the QG model with the help of text evidence. We conduct experiments on two datasets SimpleQuestions and WebQSP with the Freebase knowledge base. Empirical results show that our framework improves the performance of KBQA and performs comparably with or even better than the state-of-the-arts.

Keywords: Question answering · Question generation · Knowledge graph

1 Introduction

Question Answering over Knowledge Base (KBQA), which allows users to ask questions in natural languages over a knowledge base, is a fundamental task of artificial intelligence and natural language processing. Generally, given a natural language question q, we can translate it into a triple $t = \langle subj, rel, obj \rangle$, where obj is the final answer while $subj$ and rel are the topic entity and relation detected from the question q. Once we find the entity and relation phrases in q and link them into entities and predicates in KB, the answers of q could be found.

One of main challenges of KBQA is it requires large-scale training data to achieve satisfying performance. Especially in the open domain scenarios, various questions asked by users may be unseen in the training process of QA model. This significantly hinders the performance of existing KBQA approaches. However, it is prohibitively expensive or even impossible to label a large-scale dataset that can cover the whole knowledge base. As relation detection is more difficult than entity linking in KBQA task [21], a QA model typically fails to answer a question because of unseen predicates or phrases. On one hand, a QA model usually tends to give a lower score to the unseen predicates. On the other hand, even if the training set contains the predicate rel, it is difficult for QA model to answer q if the corresponding paraphrases are unseen.

Question Generation (QG) can be regarded as a reverse task of QA, which generates a corresponding question q given the answer a. In different QA/QG tasks the answer a can be different such as a sentence in a document or a fact in knowledge base. Inspired by the success of leveraging Question Generation (QG) to help reading comprehension [16] and answer sentence selection [14] tasks, we attempt to improve the performance of KBQA by employing the factoid QG.

In this work, we propose a unified framework to combine QA and QG through two components including dual learning and fine tuning. Similar with [14], we first train the models of QA and QG jointly by utilizing the probabilistic correlation between them. As the answer a is a sentence in [14] but a triple in our KBQA task, we design different methods to calculate the corresponding terms in the probability formula. To solve the challenges of unseen predicates and phrases, we propose a fine tuning component. By utilizing the copy action [10] and text evidence from Wikipedia, we train a sequence-to-sequence model that can generate questions of unseen predicates based on the extracted triples from knowledge base. Further, the QA model could be fined tuned by feeding the generated questions and the extracted triples from KB.

Our contribution is three-fold. First, different from previous works on reading comprehension or answer sentence selection tasks, we study how to help KBQA task by utilizing the factoid QG. Second, the fine tuning component in our framework can solve the challenges of unseen predicates and phrases in KBQA task. Third, empirical results show that the KBQA system improved by our framework performs comparably with or even better than the state-of-the-arts.

2 Our Approach

In this section, we first formulate the task of QA and QG, and then present our combination framework which utilizes QG to improve QA performance.

This work involves two tasks including question answering (QA) and question generation (QG). In natural language processing community, QA tasks can be categorized into Knowledge based and Text based. The answer in KBQA is a fact from the knowledge base while the answer in Text QA is a sentence from the given document. In this work, we focus on KBQA [7,17] and consider it as a scoring and ranking problem, formulated by $f_{qa}(q, a)$, where q is the given question and a is a triple $\langle s, p, o \rangle$ in the KB. The function outputs a scalar to estimate the relevance between q and a. For convenience, we reduce the QA task to a relation detection task, which takes a question q and candidate relations R

$= \{r_1, r_2, ..., r_n\}$ as input, and outputs a relation $r_i \in R$ which has the largest probability to be the correct relation p. In other words, we suppose the topic entity s has already been detected. Once the relation p is confirmed, we could easily obtain the answer fact a by querying KB using s and p. The task of QG takes a sentence or fact a as input, and outputs a question q which could be answered by a. In this work, we regard QG as a generation problem and develop a sequence-to-sequence model to solve it. Our QG model is abbreviated as $P_{qg}(q|a)$, of which the output is the probability of generating a question q.

Generally, our framework consists of two components. The first is dual learning component, which tries to lead the parameters of QA/QG models to a more suitable direction in training process by utilizing the probabilistic correlation between QA and QG. The second is fine tuning component, aiming to enhance the ability of QA model to tackle the unseen predicates and phrases by involving the QG model with textual corpus and KB triples. Our framework is flexible and does not rely on specific QA or QG models.

2.1 Dual Learning

Recent work [14] proposes a dual learning framework to jointly considering question answering (QA) and question generation (QG) by leveraging the probabilistic correlation between QA and QG as the regularization term to improve the training process of both tasks. The intuition is that QA-specific signals could enhance the QG model to generate not only literally similar question strings, but also the questions that could be answered by the answer. In turn, QG could improve QA by providing additional signals which stands for the probability of generating a question given the answer. The training objective is to jointly learn the QA model parameterized by θ_{qa} and the QG model parameterized by θ_{qg} by minimizing their loss functions subject to the following constraint.

$$P_a(a)P(q|a;\theta_{qg}) = P_q(q)P(a|q;\theta_{qa}) \tag{1}$$

Specifically, given a correct $\langle q, a\rangle$ pair, QA and QG models should minimize their original loss function as well as the following regularization term:

$$\begin{aligned} l_{dual}(a, q; \theta_{qa}, \theta_{qg}) = [&\log P_a(a) + \log P(q|a;\theta_{qg}) \\ &- \log P_q(q) - \log P(a|q;\theta_{qa})]^2 \end{aligned} \tag{2}$$

where $P_a(a)$ and $P_q(q)$ represent the marginal possibility of the sentence a and q, which can be calculated by the language models. While $P(q|a;\theta_{qg})$ and $P(a|q;\theta_{qa})$ represent the conditional possibility, which can be calculated by the QA model and QG model, respectively.

However, the answer a in KBQA task is a fact rather than a sentence. It is impossible to calculate $P_a(a)$ by utilizing the language model directly. To solve this problem, we propose three methods.

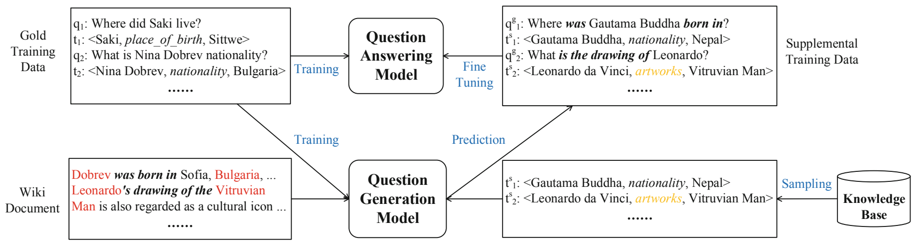
Gold Training Data
q1: Where did Saki live?
t1: <Saki, place_of_birth, Sittwe>
q2: What is Nina Dobrev nationality?
t2: <Nina Dobrev, nationality, Bulgaria>
Training
Question Answering Model
Fine Tuning
qg1: Where was Gautama Buddha born in?
ts1: <Gautama Buddha, nationality, Nepal>
qg2: What is the drawing of Leonardo?
ts2: <Leonardo da Vinci, artworks, Vitruvian Man>
Supplemental Training Data
Training
Prediction
Wiki Document
Dobrev was born in Sofia, Bulgaria, ...
Leonardo's drawing of the Vitruvian Man is also regarded as a cultural icon ...
Question Generation Model
ts1: <Gautama Buddha, nationality, Nepal>
ts2: <Leonardo da Vinci, artworks, Vitruvian Man>
Sampling
Knowledge Base

Fig. 1. The fine tuning component of our unified framework (we abbreviate the predicates for simplicity)

– **predicate frequency**. As $P_a(a)$ represents the marginal possibility of a, the most straightforward idea is to simulate it using the frequency[1] of a. In this task, we can regard all triples in the training data as the sample space. However, the frequency of each triple in the dataset typically has no significant difference, since the dataset organizer would like to cover more entities and predicates which leads to less repetitive triples. On the other hand, we find that predicates plays a more important role than subject and object in the inference process of QA/QG models. Therefore, we utilize the frequency of predicates to represent $P_a(a)$.
– **translate by templates**. [14] employs language models to calculate the relative likelihood of question q and answer a since they are both natural languages. Thus another solution to obtaining $P_a(a)$ is to translate the triple a into a natural language sentence s_a and then utilize the pre-trained language model to calculate the probability of s_a. To translate the triple $a = \langle subj, rel, obj\rangle$ to a sentence s_a, we first try a template-based method. As most KB predicates represent equivalent meanings with their word representation, we can split the predicate rel to a sequence of words and utilize it to construct the sentence s_a according to predefined templates.
– **translate by NAG model**. To improve the diversity of translated questions, we try to translate the triple a to sentence s_a by utilizing a pretrained Natural Answer Generation model [9].

2.2 Fine Tuning

Training KBQA systems relies on high-quality annotated datasets that are not only large-scale but also unbiased. However, it is difficult to build such a dataset which covers equally a large number of triples in the knowledge base.

Therefore, we propose a fine tuning framework to supplement the QA dataset and improve the capacity of QA models. Figure 1 shows the framework. We first train the QA model and QG model using the whole training set $T = \{(q_1, a_1), (q_2, a_2), ..., (q_n, a_n)\}$.For each triple $a_i = \langle subj, rel, obj\rangle$, we collect a set of textual evidence from wiki documents to help the training and inference of QG model.

[1] According to the Law of Large Numbers, the frequency can represent the probability if the sample space is large enough.

Collecting Textual Evidence. Following the distant supervision setup for relation extraction [11], we first select the sentences containing both the subject *subj* and the object *obj* from the Wikipedia articles of the entity *subj*. Then those sentences are reduced to relation paraphrases by reserving the words those appear on the dependency path between *subj* and *obj*. We collect the list of entity types of *subj* and *obj* by querying the knowledge base. If an entity has multiple types we pick the type which occurs in the selected sentence s or the predicate *rel*. Finally we replace the *subj* and *obj* mentions with their types to learn a more general relation representation in syntactic level. In Fig. 1, a possible textual evidence generated from the sentence "Dobrev was born in Sofia, Bulgaria, ..." is "*Person* was born in *Country*". With the help of text evidence, the QG model is able to generate questions for unseen predicates.

After the normal training process[2], we build a set of supplemental question-answer pairs to fine tune the QA model. Specifically, we sample a set of triples from the knowledge base and collect the text evidences of these triples from the Wiki documents. Then the QG model generates corresponding questions by feeding the triples and text evidences. We can regard the generated questions and the sampled triples as the supplemental training set. The remaining problem is how to sample the triples from the knowledge base. Intuitively, the more triples we sample from knowledge base the better capacity can be enhanced of QA model. However, the total number of triples in KB is too large, it is necessary to study how to sample appropriate triples, as described in the following.

Sampling KB Triples. The straightforward strategy is to select triples randomly. We first obtain the candidate predicate set R containing predicates with top k frequencies. Then we select predicate m times from R. For each selected predicate rel_i, we query the KB to find a corresponding pair of subject $subj_i$ and obj_i randomly, after that we get a triple $\langle subj_i, rel_i, obj_i \rangle$. Finally the supplemental triple set T is built completely when it has m triples, where m is a hyper parameter. To avoid tuning the parameter m, we propose a method to sample an unbiased triple set with the same distribution of the original data set. As a premise, we suppose the test set has the same distribution with knowledge base while has a little difference with the training set. In order to supplement the training set, we create a predicate set R by random selecting. The selecting process is terminated when each predicate rel_i in the original training set has occurred in R. After that we discard all these redundant predicates and regard the remaining predicates as the supplemental predicate set.

Example 1. Consider Fig. 1. The models of QA and QG are first trained by the original training set $\{(q_1, t_1), (q_2, t_2)\}$. During the training, QG model learns how to generate questions utilizing the copy action and the text evidence extracted from the wiki documents. As the predicate <artworks> is unseen, the QA model can not answer the questions like "q^t=What is the drawing of [subj]" with the answer triple $\langle subj, artworks, obj \rangle$. However after fine tuning, the QA model with the sampled triple $t_2^s = \langle$Leonardo da Vinci, artworks, Vitruvian Man$\rangle$ and generated question q_2^g, can answer q^t correctly. On the other hand, it is hard for

[2] Note that in this process the QA and QG models could be trained utilizing the dual learning framework.

QA model to link the question "Where was [subj] born in" to the gold predicate <nationality> because the phrase is unseen in the training phase. While fine tuning can bring such unseen phrases to enhance the capacity of the QA model.

3 Models

3.1 QA Model

We describe the details of the question answering (QA) model in this section. Generally, a QA model could be formulated as a function $f_{qa}(q, a)$ that estimates the correctness of every candidate answer a given the question q. For convenience, we reduce the QA model to a relation classification model and use the candidate predicate rel to replace the answer a. Compared with other subtasks such as entity linking in KBQA, relation extraction plays a more significant role in affecting the final results [21]. The accuracy of entity linking are relatively high in existing KBQA methods while the performance of relation extraction is not good enough due to the unseen predicates or paraphrases.

We propose a simple yet effective relation extraction model based on recurrent neural network (RNN). To better support the unseen relations, we factorize the relation names to word sequences and formulate relation extraction as a sequence matching and ranking task.Specifically, the input relation becomes $\mathbf{r} = \{r_1, ..., r_m\}$, where the m tokens are split into relation names. For example, the relation *location.country.languages_spoken* can be divided into {*location, country, languages, spoken*}. Each token above is transformed to its pre-trained word embedding [12] then we use a Bidirectional Long Short-Term Memory (BiLSTM) [22] to obtain the hidden representations. A max pooling layer is employed to extract the most salient local features to form a fixed-length global feature vector, then we obtain the final relation representation $\mathbf{h}^r$.

We use the same neural network to get the question representation $\mathbf{h}^q$ and then compute the similarity using cosine distance function. To learn a more general representation in the syntactic level, we replace the entity mention with a generic symbol <e>, such as "where is <e> from". However, this mechanism discards all entity information and might confuse the model in some cases. Therefore, we detect the type t of topic entity and concatenate the type representation with question representation. We find that this type information could improve the performance significantly.

The model described above is trained with a ranking training approach, which drives the model to output a high score for question q with gold relation r^+ while producing a lower score for incorrect relations r^- in the candidate relation pool R. The loss function is denoted as following.

$$l_{rel} = max\{0, \lambda - S_s(q, r^+) + S_s(q, r^-)\} \tag{3}$$

where the pair of question and correct predicate (q, r^+) are forced to have a score of at least margin λ and $S_s(q, r) = Cosine(\mathbf{h}^q, \mathbf{h}^r)$. The candidate relation pool R consists of all predicates connected with the gold topic entity e in q.

3.2 QG Model

Factoid QG is the task generating natural language questions given an input triple from knowledge bases. The generated question is concerned with the subject and predicate of the fact, and the object of the fact represents a valid answer to the generated question [13]. The QG model approximates the conditional probability of the generated question $q = \{w_1, w_2, ..., w_n\}$ given an input fact $a = \{s, p, o\}$, formulated as:

$$p(q|a) = \prod_{t=1}^{n} p(w_t|w_{<t}, a) \tag{4}$$

where $w_{<t}$ denotes all previous generated words until time step t. Inspired by the recent success of sequence-to-sequence learning in Neural Machine Translation [1], we treat the QG problem as a kind of translation task and employ the encoder-decoder architecture to tackle it. Specifically, the encoder encodes the given fact $a = \{s, p, o\}$ into three fixed size vectors $h_s = \mathbf{E_f} e_s$, $h_p = \mathbf{E_f} e_p$ and $h_o{=}\mathbf{E_f} e_o$, where $\mathbf{E_f}$ is the KB embedding matrix learned using TransE [3], e_s, e_p and e_o are one-hot vectors of s, p and o. We concatenate those three vectors to obtain the encoded fact $h_f = [h_s; h_p; h_o]$. Later the decoder takes h_f to generate a question in a sequential way.

Note that in the fine tuning component, we leverage the QG model to generate supplemental question-answer pairs to fine tune the trained QA model. We expect the supplemental labeled data to contain the predicates or phrases not encountered by the QA model during training process so that the QA model can enhance its capability. Thus the QG model should be able to generate questions given the triples with unseen predicates. Following [6], we introduce a text encoder. For each fact a we collect n textual evidence $D = \{d_1, d_2, ..., d_n\}$ from wiki documents. A set of n Gated Recurrent Neural Networks (GRU) with shared parameters are utilized to encode each textual evidence. The hidden state of i-th word in j-th textual evidence is calculated as:

$$h_i^{d_j} = GRU_j(\mathbf{E_d} w_i^j, h_{i-1}^{d_j}) \tag{5}$$

where $\mathbf{E_d}$ is the pre-trained word embedding matrix [12] and w_i^j is the one-hot vector of i-th word in d_j. We concatenate each hidden state of textual evidence to get the final encoded text $h_d = [h_{|d_1|}^{d_1}; h_{|d_2|}^{d_2}; ...; h_{|d_n|}^{d_n}]$.

For the decoder we use a GRU with an attention mechanism [1] acting over the input textual evidence. Given a set of encoded input vectors $I = \{h_1, h_2, ..., h_k\}$ and the decoder's previous hidden state s_{t-1}, the attention mechanism calculates $\alpha_t = \{\alpha_{i,t}, ..., \alpha_{k,t}\}$ as a vector of scalar weights, each $\alpha_{i,t}$ determines the weight of its corresponding encoded input vector h_i.

$$e_{i,t} = \mathbf{v_a}^{\top} tanh(\mathbf{W_a s_{t-1}} + \mathbf{U_a h_i}) \tag{6}$$

$$\alpha_{i,t} = \frac{exp(e_{i,t})}{\sum_{j=1}^{k} exp(e_{j,t})} \tag{7}$$

where $\mathbf{v_a}$, $\mathbf{W_a}$, $\mathbf{U_a}$ are trainable weight matrices of the attention modules. Then we calculate an overall attention over all tokens in all textual evidence:

$$a_t^d = \sum_{j=1}^{|D|} \sum_{i=1}^{|d_i|} \alpha_{i,t}^{d_j} h_i^{d_j} \tag{8}$$

where $\alpha_{i,t}^{d_j}$ is a scalar value determining the weight of the i-th word in the j-th textual evidence d_i at time step t.

Recent works on NMT tackle the rare/unknown words problem using copy actions [10]. It copies the words with a specific position from the source to the output text. We leverage this mechanism to solve the issue of unseen predicates. We adopt a variant of [6] which copies the words with same POS tags rather than specific positions. This can improve the generalization ability of our QG model. At each time step, the decoder chooses to output either a word from the vocabulary or a special token indicating a copy action from the textual evidence. Those special tokens are replaced with their original words before being outputted.

4 Experiment

4.1 Setup

We conduct experiments on two datasets SimpleQuestions [2] and WebQSP [18]. Each question in these datasets is labeled with the gold semantic parse so that we can evaluate both relation detection task with gold entity linking results and the KBQA task independently.

SimpleQuestions (SQ) is a large scale KBQA dataset with more than 100 thousand labeled data. Each question in SQ has only one entity and one relation, which can be answered by a single triple in knowledge base. We use the Freebase subset with 2M entities (FB2M) [2] in order to compare with previous works. For relation detection task, we use the dataset processed by [19] for comparison.

WebQSP is a medium scale KBQA dataset containing both single-triple and multi-triple questions. Following [18], we use S-MART [15] entity-linking outputs. For relation detection task, we use the dataset processed by [21] for comparison.

4.2 Relation Detection Results

Table 1 shows the relation detection accuracy when using different percentages of gold data to train the models. QA Baseline is the model described in Sect. 3.1. Dual Learning trains QA and QG models simultaneously and improves the performance on both two datasets. To further demonstrate the effectiveness of fine tuning component, we run the entire pipeline (Dual Learning + Fine Tuning) with different percentages of training data. When the ratio is 50%, we randomly select 50% question-triple pairs from training data as available part and regard the other 50% as sampling part. The QA and QG models are trained jointly by feeding the available part. For each triple in the sampling part we use QG model to generate the supplemental training data and then tune the QA model. When the ratio is 100%, we use the method in Sect. 2.2 to sample the extra triples from KB and generate the supplemental training data.

Table 1. Relation detection accuracy in different ratios of available gold training data

Methods	SQ			WebQSP		
	5%	50%	100%	5%	50%	100%
BiCNN [17]	–	–	90.0	–	–	77.74
AMPCNN [19]	–	–	91.3	–	–	–
HR-BiLSTM [21]	–	–	**93.3**	–	–	82.53
QA baseline	88.3	91.0	91.9	51.76	72.95	80.56
Dual learning	88.7	91.5	92.7	52.64	74.53	81.87
Dual learning + Fine tuning	89.8	91.7	93.0	54.37	79.02	**83.63**

The fine tuning component improves the accuracy on all levels of available gold data. For WebQSP, the largest increase (4.49%) occurs in WebQSP-50%. This is because using 50% training data leads to more unseen predicates and larger improvement margin than using 100% training data. Although WebQSP-5% has most unseen predicates, such a small number (155) of training data limits the generalization ability of QG model. For SQ the largest increase (1.1%) occurs in SQ-5% rather than SQ-50% since the former already has enough training data (3611). When using 100% gold training data, the accuracy improvement on WebQSP is larger than on SQ. The underlying reason is that SQ has too many repetitive predicates and a very small number (0.7%) of unseen predicates, i.e., the improvement space of SQ dataset is relatively small.

We also compare our framework with existing QA methods.BiCNN model is re-implemented by [21] from STAGG [17], where both questions and relations are represented with the word hash trick on character tri-grams, and we report their results directly. AMPCNN [19] propose an attentive max pooling stacking over word-CNN, so that the predicate representation can be matched with the predicate-focused question representation more effectively. HR-BiLSTM [21] propose a hierarchical recurrent neural network enhanced by residual learning to compare questions and relations via different levels of abstraction and achieves state-of-the-art results for both SimpleQuestions and WebQSP datasets. Our entire pipeline (83.63%) outperformed the state-of-art result (82.53%) on WebQSP while still having a minor gap (0.3%) to reach the state-of-art on SimpleQuestions. Note that the final results could be improved by refining our simple QA model using more complex neural network architectures, we leave it as future work.

4.3 Comparison Results of Dual Learning

Table 2 shows the relation detection results using different methods to calculate the marginal possibility $P_a(a)$ in the formula 1. Dual Learning with templates based translation achieves the best performance among all these methods. Simulating $P_a(a)$ to the predicate frequency performs poor (79.1%) on WebQSP dataset. This is most likely because the sample space of a, i.e., the training data size is too small. It is interesting that template-based translation method has better performance than NAG method. Although translating the triples according to templates has lower diversity and fluency than utilizing the sequence-to-sequence model, the latter one may be hard to learn with a small-scale supervised dataset.

Table 2. Comparison results of dual learning (Accuracy)

	$P_a(a)$ calculation	SQ	WebQSP
Dual learning	Predicate frequency	91.6	79.1
Dual learning	Translate by templates	**92.7**	**81.9**
Dual learning	Translate by NAG	92.1	80.5

4.4 QG Performance

Table 3 describes the performance of QG model trained with 100% training data. To evaluate the correctness we sample 100 generated questions from the test set. A question q is regarded as correct if it represents the target predicate (no matter of the fluency). The results show that our QG model is able to generate supplemental training data in fine tuning component with high quality.

Table 3. Results of QG model, the correct ratio is evaluated by human on 100 sampled questions

	BLEU-4	Correct ratio
SimpleQuestions	34.6	94%
WebQSP	39.0	93%

4.5 KBQA End-Task Results

Finally we evaluate the end-to-end performance on KBQA task. The accuracy of KBQA end task is shown in Table 4. Our approach performs better (64.4%) than the state-of-art systems (63.9%) on WebQSP while still having a minor gap (0.2%) to reach the state-of-art on SQ.Our KBQA pipeline is similar with [21], we use entity linking outputs from S-MART [15] and AMPCNN [19] for WebQSP and SimpleQuestions.

Table 4. Results of question answering

	SimpleQuestions	WebQSP
STAGG [17]	72.8	63.9
AMPCNN [19]	76.4	–
HR-BiLSTM [21]	**78.7**	63.9
Our approach	78.5	**64.4**

4.6 Case Study

Table 5 lists some examples to compare the QA baseline and fine tuned QA. The baseline predicts an incorrect relation <music.group_member.instruments_played> of question q due to gold relation r = <music.group_membership.role> is absent in the training process. Given a sampled triple containing this unseen relation with textual evidence, QG model generates a question q' having similar hidden representation with q. After fine tuning by feeding (q', r), the QA model can predict correctly.

Table 5. Examples of the fine tuning framework.

Gold question (with entity type)	Gold Triple	Prediction (Baseline)
q : what role did Paul McCartney in the Beatles? (Musical Artist)	<Paul_McCartney, music.group_membership.role, Lead_Vocals>	<Paul_McCartney, music.group_member.instruments_played, Guitar>
Generated question (with entity type)	Sampled triple	Prediction (Fine Tuned)
q' : what role does John Lennon play? (Musical Artist)	<John_Lennon, music.group_membership.role, Drums>	<Paul_McCartney, music.group_membership.role, Lead_Vocals>

5 Related Work

There are different types of QA tasks including text based QA [20] and knowledge based QA [8]. Our work belongs to knowledge based QA where the answer are facts in KB. Yu et al. [21] design a neural relation detection model to improve the question answering performance. They use deep residual bidirectional LSTMs to compare questions and relation names via different levels of abstraction and achieve state-of-the-art accuracy for SimpleQuestions and WebQSP datasets. Hu et al. [8] propose a state-transition framework to parse the questions into complex query graphs utilizing several predefined operations. Dong et al. [4] train a sequence to tree model to translate natural language to logical forms.

Question Generation draws a lot of attentions in many applications. Luong et al. [10] propose a model that generates positional placeholders pointing to some words in source sentence and copy it to target sentence, i.e., the copy actions. Dong et al. [5] generate paraphrases of given questions to increase the performance of QA systems which rely on paraphrase datasets, neural machine translation and rule mining. ElSahar et al. [6] present a neural model for factoid QG in a Zero-Shot setup, that is generating questions for triples containing predicates, subject types or object types that were not seen at training time.

6 Conclusion

In this paper we study how to utilize Question Generation (QG) models to help Knowledge Base Question Answering (KBQA). Specifically, we propose a unified framework to combine QA and QG with the help of knowledge base and text corpus. The models of QA and QG are first trained jointly on the gold dataset by utilizing the probabilistic correlation between them, then the QA model is fine tuned by utilizing a supplemental dataset constructed by the QG model with the help of text evidence. The proposed framework can solve the challenges of unseen predicates and phrases in KBQA. Empirical results show that our framework improves the performance of KBQA and performs comparably with or even better than the state-of-the-arts.

Acknowledgments. This work was supported by The National Key Research and Development Program of China under grant 2018YFB1003504 and NSFC under grant 61961130390, 61622201 and 61532010.

References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. CoRR abs/1409.0473 (2014)
2. Bordes, A., Usunier, N., Chopra, S., Weston, J.: Large-scale simple question answering with memory networks. CoRR abs/1506.02075 (2015)
3. Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Proceedings of NIPS (2013)
4. Dong, L., Lapata, M.: Language to logical form with neural attention. In: Proceedings of ACL (2016)
5. Dong, L., Mallinson, J., Reddy, S., Lapata, M.: Learning to paraphrase for question answering. In: Proceedings of EMNLP, pp. 875–886 (2017)
6. ElSahar, H., Gravier, C., Laforest, F.: Zero-shot question generation from knowledge graphs for unseen predicates and entity types. In: Proceedings of NAACL-HLT, pp. 218–228 (2018)
7. Hu, S., Zou, L., Yu, J.X., Wang, H., Zhao, D.: Answering natural language questions by subgraph matching over knowledge graphs. Trans. Knowl. Data Eng. **30**(5), 824–837 (2018)
8. Hu, S., Zou, L., Zhang, X.: A state-transition framework to answer complex questions over knowledge base. In: Proceedings of EMNLP, pp. 2098–2108 (2018)
9. Liu, C., He, S., Liu, K., Zhao, J.: Curriculum learning for natural answer generation. In: Proceedings of IJCAI, pp. 4223–4229 (2018)

10. Luong, T., Sutskever, I., Le, Q.V., Vinyals, O., Zaremba, W.: Addressing the rare word problem in neural machine translation. In: Proceedings of ACL (2015)
11. Mintz, M., Bills, S., Snow, R., Jurafsky, D.: Distant supervision for relation extraction without labeled data. In: Proceedings of ACL, pp. 1003–1011 (2009)
12. Pennington, J., Socher, R., Manning, C.D.: GloVe: global vectors for word representation. In: Proceedings of EMNLP, pp. 1532–1543 (2014)
13. Serban, I.V., et al.: Generating factoid questions with recurrent neural networks: the 30 m factoid question-answer corpus. In: Proceedings of ACL (2016)
14. Tang, D., Duan, N., Qin, T., Zhou, M.: Question answering and question generation as dual tasks. CoRR abs/1706.02027 (2017)
15. Yang, Y., Chang, M.: S-MART: novel tree-based structured learning algorithms applied to tweet entity linking. In: Proceedings of ACL, pp. 504–513 (2015)
16. Yang, Z., Hu, J., Salakhutdinov, R., Cohen, W.W.: Semi-supervised QA with generative domain-adaptive nets. In: Proceedings of ACL, pp. 1040–1050 (2017)
17. Yih, W., Chang, M., He, X., Gao, J.: Semantic parsing via staged query graph generation: question answering with knowledge base. In: ACL (2015)
18. Yih, W., Richardson, M., Meek, C., Chang, M., Suh, J.: The value of semantic parse labeling for knowledge base question answering. In: Proceedings of ACL (2016)
19. Yin, W., Yu, M., Xiang, B., Zhou, B., Schütze, H.: Simple question answering by attentive convolutional neural network. In: COLING, pp. 1746–1756 (2016)
20. Yu, L., Hermann, K.M., Blunsom, P., Pulman, S.: Deep learning for answer sentence selection. CoRR abs/1412.1632 (2014)
21. Yu, M., Yin, W., Hasan, K.S., dos Santos, C.N., Xiang, B., Zhou, B.: Improved neural relation detection for knowledge base question answering. In: Proceedings of ACL (2017)
22. Zhou, P., et al.: Attention-based bidirectional long short-term memory networks for relation classification. In: Proceedings of ACL (2016)