Knowing What I Don't Know: A Generation Assisted Rejection Framework in Knowledge Base Question Answering

Junyang Huang[1], Xuantao Lu[1], Jiaqing Liang[1], Qiaoben Bao[1], Chen Huang[3], Yanghua Xiao[12(✉)], Bang Liu[4], and Yunwen Chen[5]

[1] Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University, Shanghai, China
[2] Fudan-Aishu Cognitive Intelligence Joint Research Center, Shanghai, China
[3] CES Finance Co., Ltd., Shanghai, China
[4] Mila & DIRO, Université de Montréal, Montréal, Québec, Canada
[5] DataGrand Inc., Shanghai, China
{jyhuang19,xtlu20,qbbao19,shawyh}@fudan.edu.cn, l.j.q.light@gmail.com, huangchen@kiiik.com, bang.liu@umontreal.ca, chenyunwen@datagrand.com

Abstract. Existing Knowledge Base Question Answering (KBQA) systems suffer from the sparsity issue of Knowledge Graphs (KG). To alleviate the issue of KG sparsity, some recent research works introduce external text for KBQA. However, such external information is not always readily available. We argue that it is critical for a KBQA system to know whether it lacks the knowledge to answer a given question. In this paper, we present a novel **Ge**neration **A**ssisted **R**ejection (GEAR) framework that identifies unanswerable questions well. GEAR can be applied to almost all KBQA systems as an add-on component. Specifically, the backbone of GEAR is a sequence-to-sequence model that generates candidate predicates to rerank the original results of a KBQA system. Furthermore, we devise a Probability Distribution Reranking algorithm to ensemble GEAR and KBQA since the architectural distinctions between GEAR and KBQA are vast. Empirical results and case study demonstrates the effectiveness of our framework in improving the performance of KBQA, particularly in identifying unanswerable questions.

Keywords: Knowledge Graph · Knowledge Base Question Answering · Natural Language Generation.

1 Introduction

Knowledge Base Question Answering (KBQA) aims to answer questions by using a large collection of triples in a Knowledge Graph (KG). Nowadays, KBQA systems are widely leveraged in lots of web applications due to the fact that a KG is in general highly accurate and contains few noisy data. Unlike many question answering tasks where the answer of a question can be retrieved from the input data, the questions for KBQA tend to be more diverse [12] and some

of them are unanswerable because the corresponding facts might be absent [7] in the KG. However, constructing a well-designed and complete KG with high precision requires unaffordable human efforts [11], which limits the coverage of a KG. Thus, more often than not, KG is incomplete to answer an open-domain question.

Building a KBQA system with the ability to identify unanswerable questions is an important but difficult task. However, there has been few research studying this problem. Introducing extra information to supply missing evidence in KG [11] and employing ensemble models with different sources are straightforward ideas to alleviate the problem of unanswerable questions. However, the gold answer may not always appear in external materials. Besides, these methods enlarge noises when retrieving relevant data. Another problem is that predicate linking in KBQA is often modeled as a binary classification task, which requires both positive and negative samples. However, the KG in KBQA can not provide all negative samples because KG is incomplete which incurs overfitting on negative samples and finally negatively impacts on the model's performance. REINFORCE algorithm based KBQA models are able to actively identify unanswerable questions, but they are limited to low accuracy on answerable questions. Godin et al. [6] proposed a ternary reward function that rewards an agent for not answering a question. However, this method realizes this idea by adding a virtual predicate "No Answer" to every entity, which leads to low recall on answerable questions.

To overcome the problems mentioned above, we propose a **Ge**neration **A**ssisted **R**ejection (GEAR) framework, which helps KBQA better identify unanswerable questions. The framework is an add-on component for KBQA systems that takes the result of a KBQA system into consideration through the following steps: **(1) Predicate Generation:** GEAR directly generates the predicate to the question through a sequence-to-sequence (seq2seq) model, i.e., "What is the highest point in Canada?" $\rightarrow$ "highest point". The seq2seq model is trained only on positive samples that overcomes the problem of overfitting on negative samples. **(2) Result Re-ranking:** GEAR re-ranks the answers of the KBQA system through the latent space of the generated predicate from the seq2seq model. Specifically, we apply a Probability Distribution Re-ranking (PDR) algorithm to assemble seq2seq model and KBQA model. The ensemble of a typical seq2seq model and a KBQA model is difficult without the PDR algorithm due to the vast distinction in the structure of seq2seq and KBQA models.

Extensive evaluations and case studies demonstrate that the performance of KBQA models improved significantly when GEAR was incorporated especially for unanswerable questions. In summary, we have made the following contributions:

- We propose a novel framework GEAR for KBQA model that improves the performance of KBQA especially on identifying unanswerable questions. To the best of our knowledge, this is the first attempt to adopt generation methods in identifying unanswerable questions.
- We devise an algorithm PDR to overcome the difficulty in implementing the ensemble of seq2seq models and typical KBQA models. The ensemble of

KBQA model and GEAR strengthens robustness and generalization capability of processing a question.
– GEAR has a broad scope of application. The framework can be easily applied to almost every KBQA model, which improves the performance of off-the-shelf KBQA systems.

2 Task Definition

The input of KBQA task is a natural language question, and the output is an answer object from KG or "No Answer" indicating that KBQA can't answer the question. More specifically, a KG is denoted as $\mathcal{K}$, which is a set of triples in the form of (s, p, o), where s, p, o denote subject, predicate and object respectively. Here, the object entity is also called answer entity. The task is, given a natural language question denoted as X containing M tokens $X = [x_1, x_2, \ldots, x_M]$ and the entity of the question, the KBQA system should retrieve the gold answer if KG $\mathcal{K}$ contains the answer. Otherwise, the KBQA system should return "No Answer" indicating no relevant information exists in KG $\mathcal{K}$.

3 Methodology

3.1 Overview of GEAR

There are two core components in GEAR: the Generation Module and the Re-ranking Module. The interaction between GEAR and KBQA model is shown in Figure 1. First, the question will be sent to a KBQA model. The KBQA model will score every candidate predicate-answer pair retrieved from KG. The score represents the confidence in each candidate predicate-answer pair. Then, the Generation Module generates a sequence probability matrix $\mathbf{C}$ and a predicate sequence in line with the question. Finally, the Re-ranking Module re-ranks the result from KBQA model based on the sequence probability matrix $\mathbf{C}$ through PDR. The output of Re-ranking Module is the final re-ranked score of every candidate predicate-answer pair. If the candidate predicate-answer pair with the highest confidence is lower than hyper-parameter τ, the system will reject answering the question.

3.2 Generation Module

The Generation Module, in this paper, aims to generate the predicate for a given input question sequence via a transformer-based encoder-decoder architecture, which consists of two components: (a) an encoder that computes a representation for each source sentence and (b) a decoder that generates one target word at each timestep since the conditional probability is decomposed into several timesteps.

Given the question token sequence $X = [x_1, x_2, \ldots, x_M]$ as input, GEAR outputs: 1) a sequence represents the predicate of the input question sequence X, which is denoted as $Y = \{y_1, y_2, \ldots, y_L\}$, where y_i is the token-id of the i-th

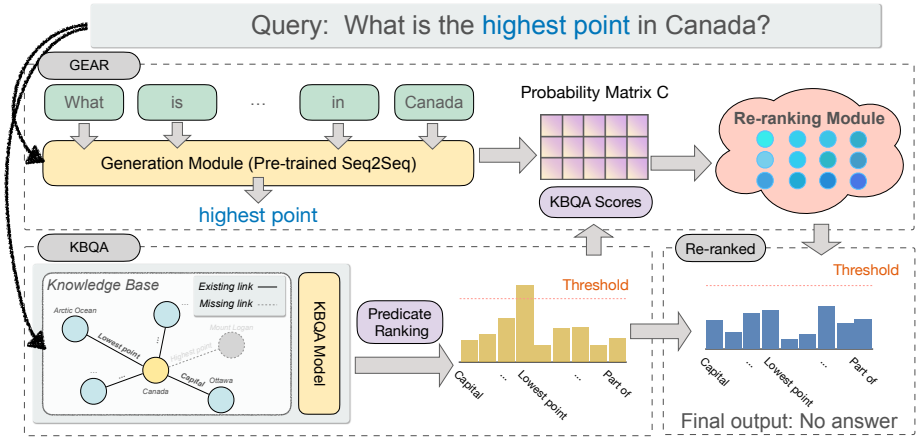

Fig. 1. The illustration of the interaction between GEAR and KBQA. GEAR is composed of a Generation Module (Section 3.2) and a Re-ranking Module (Section 3.3). GEAR re-ranks the candidate answers retrieved by the KBQA model.

word in Y. M and L are the lengths of input sequence and output sequence respectively. 2) a sequence probability matrix $\mathbf{C}$. The sequence probability matrix is the probability distribution of each candidate character in every step of the generated sequence, which is denoted as $\mathbf{C} = \{\mathbf{c}_1, \ldots, \mathbf{c}_L\}$, L is the length of the sequence. $\mathbf{c}_i = \{c_{i1}, \ldots, c_{iW}\}$ is the character probability distribution of the i-th decoding step, c_{ij} is the j-th token, W is the length of the vocabulary list $\mathcal{V}$. The sequence probability matrix $\mathbf{C}$ will be used as part of the input of the Re-ranking Module.

We adopt the pre-trained language model BART [8] as our transformer-based seq2seq backbone. In this way, the general semantic knowledge of natural language and text generation knowledge can be directly reused. The encoder of the Generation Module first learns the hidden states $\mathbf{H} = \mathbf{h}_i, \ldots, \mathbf{h}_{|M|}$ of the given input question sequence through a multi-layer transformer encoder:

$$\mathbf{H} = \text{Encoder}(x_1, x_2, \ldots, x_M), \tag{1}$$

where each layer of Encoder$(\cdot)$ is a transformer block with the multi-head attention mechanism. After the input question token sequence is encoded, the decoder predicts the output predicate token-by-token with the sequential input tokens' hidden vectors. At the timestep i of generation, the self-attention decoder predicts the i-th token p_i in the linearized form and decoder hidden state $\mathbf{h}_i^d$ as:

$$y_i, \mathbf{h}_i^d = \text{Decoder}([\mathbf{H}; \mathbf{h}_1^d, \ldots, \mathbf{h}_{i-1}^d], y_{i-1}), \tag{2}$$

where each layer of Decoder$(\cdot)$ is a transformer block that includes self-attention with decoder state $\mathbf{h}_i^d$ and cross-attention with encoder state $\mathbf{H}$. The generated output predicate begins with "<s>" and ends with the end token "</s>".

Algorithm 1 Probability Distribution Re-ranking

Input: Sequence probability matrix $\mathbf{C}$ from GEAR, candidate predicate sequence $\mathcal{P}$, the score of the candidate predicate $\mathcal{S}_\mathcal{P}$ from KBQA, the weight parameter η, the length of the generated predicate L and the length of the candidate predicate s.
Output: The score $\mathcal{S}$ of the candidate predicate sequence $\mathcal{P}$.

$k \leftarrow 0, \mathcal{S} \leftarrow 0$
if $L \leq s$ **then**
  while $k \leq s - L$ **do**
    $\mathcal{S} \leftarrow \max(\sum_{i=1}^{l} c_{i+k,p_i}, \mathcal{S})$
    $k \leftarrow k + 1$
  end while
else
  while $k \leq L - s$ **do**
    $\mathcal{S} \leftarrow \max(\sum_{i=1}^{s} c_{i,p_{i+k}}, \mathcal{S})$
    $k \leftarrow k + 1$
  end while
end if
$\mathcal{S} \leftarrow \frac{\eta \mathcal{S}}{max(s,L)} + (1 - \eta)\mathcal{S}_\mathcal{P}$
return $\mathcal{S}$

The probability of each step $p(y_i|y_{<i}, x)$ is gradually added to the conditional probability of the entire output sequence $p(y|x)$:

$$p(y|x) = \prod_{i}^{|y|} p(y_i|y_{<i}, X), \tag{3}$$

where $y_{<i} = y_1, \ldots, y_{i-1}$, and $p(y_i|y_{<i}, x)$ is the probability over the target vocabulary list $\mathcal{V}$ normalized by softmax function. At step i, $\mathbf{c}_i$ represents the normalized probability distribution over $\mathcal{V}$. The objective function of GEAR during training is to minimize the negative log-likelihood loss, θ represents the parameters of the model:

$$\mathcal{L} = -\sum_{t=0}^{L} \log P_\theta(y_t|y_{<t}, X), \tag{4}$$

3.3 Re-ranking Module

The goal of the Re-ranking Module is to assign new scores for matched predicates in KG retrieved by KBQA model. Specifically, the PDR algorithm re-ranks the score with the generated results from the Generation Module. PDR is a re-ranking algorithm that re-ranks the result of KBQA with the help of the sequence probability matrix $\mathbf{C}$ generated by the Generation Module. The input to PDR is a sequence probability matrix $\mathbf{C}$, a candidate predicate sequence $\mathcal{P} = \{p_1, p_2, \ldots, p_s\}$, the score $\mathcal{S}_\mathcal{P}$ of the candidate predicate $\mathcal{P}$ calculated by

Table 1. Evaluation of different KBQA models on SimpleQuestions and NLPCCKBQA datasets with a varying fraction of KG-size. "+Gear" means the result is re-ranked by Gear. The best results in each group are highlighted in bold.

SimpleQuestions	**100%**	**70%**	**50%**	**30%**	**10%**
BiLSTM	80.3 / –	79.9 / 65.0	77.9 / 68.5	79.7 / 67.5	**82.1** / 66.8
BiLSTM+Gear	**80.5** / –	**80.2** / **76.3**	77.9 / **77.2**	79.7 / **76.2**	79.2 / **75.4**
MCCNN	**79.8** / –	**79.4** / 71.4	**78.7** / 71.9	79.5 / 71.6	76.8 / 71.5
MCCNN+Gear	79.5 / –	79.2 / **77.2**	77.6 / **76.1**	**79.7** / **76.6**	**77.4** / **76.6**
BAMnet	**82.1** / –	**81.4** / 62.0	80.1 / 66.4	**81.5** / 65.0	**83.9** / 64.3
BAMnet+Gear	81.9 / –	80.8 / **73.1**	**80.8** / **74.4**	81.2 / **73.5**	83.3 / **72.8**
BBKBQA	88.3 / –	**88.4** / 68.0	88.8 / 67.0	87.4 / 66.4	**90.5** / 66.2
BBKBQA+Gear	**88.4** / –	88.2 / **72.3**	**89.3** / **71.2**	**87.8** / **71.3**	89.2 / **71.1**
NLPCCKBQA	**100%**	**70%**	**50%**	**30%**	**10%**
BiLSTM	66.8 / –	66.7 / 65.0	65.5 / 65.4	66.3 / 64.7	65.3 / 64.6
BiLSTM+Gear	**75.7** / –	**75.6** / **70.2**	**75.1** / **70.6**	**73.6** / **69.9**	**75.9** / **70.1**
MCCNN	77.8 / –	77.9 / 83.2	76.3 / 84.1	76.9 / 83.8	77.8 / 83.6
MCCNN+Gear	**80.8** / –	**81.0** / **87.9**	**79.3** / **88.7**	**80.0** / **88.3**	**81.1** / **88.0**
BAMnet	71.6 / –	71.9 / 65.2	70.5 / 64.4	70.4 / 64.0	69.6 / 64.4
BAMnet+Gear	**79.3** / –	**79.2** / **73.5**	**78.6** / **72.3**	**78.7** / **72.4**	**78.3** / **72.3**
BBKBQA	90.3 / –	90.6 / 81.3	89.5 / 81.9	89.7 / 81.6	89.7 / 81.1
BBKBQA+Gear	**91.5** / –	**91.9** / **83.2**	**91.1** / **83.5**	**90.8** / **83.3**	**91.1** / **82.7**

KBQA model and the weight parameter η. The output is the re-ranked final score $\mathcal{S}$ that denotes the confidence of the predicate sequence $\mathcal{P}$. The overall re-ranking procedure is described in Algorithm 1. We set η as the weight parameter that indicates the proportion of Gear in the final scoring. In other words, the proportion of KBQA in the final scoring is $1 - \eta$.

4 Experiments

4.1 Datasets, Baselines and Training Details

We evaluate our framework on two popular KBQA datasets SimpleQuestions[1] and NLPCCKBQA [4]. The two datasets are modified with 5 different KG-size setups: 100%, 70%, 50%, 30%, 10%. If KG-size parameter is set to 70%, we drop triples related to 30% random selected questions in test sets. Thus, 70% questions are answerable while the other 30% questions are unanswerable. We take four widely used KBQA models as our baselines: BiLSTM [10], MCCNN [3], BAMnet [2], and BBKBQA [9]. We do the best to adjust the hyperparameters of the baseline KBQA models so that the baseline models have relatively optimal performance. We implement the Generation Module in Gear based on the public

Table 2. Evaluation of the predicates generated by the Generation Module on SimpleQuestions and NLPCCKBQA.

Metrics	SimpleQuestions	NLPCCKBQA
EM	86.7	70.9
BLEU-1	90.7	83.5
ROUGE-L	91.1	85.4
HCI	95.8	88.4

HuggingFace implementation of the BART seq2seq model and optimize it by Adam. The initial learning rate is set to 5e−5 and we set the batch size to 32. We set $\tau = 0.5$ because re-ranking is essentially implementing binary classification for each predicate-answer pair. We set η to 0.3 based on the overall accuracy of the model for all questions on the devset.

4.2 Performance and Analysis

Evaluation of KBQA model after GEAR is applied The main results are shown in Table 1. Each grid has two results separated by "/" indicating accuracy of answerable questions and the rejection rate of unanswerable questions respectively. "-" is a placeholder since no unanswerable question exists when KG-size is 100%. The rate of rejection on unanswerable questions achieves significant improvement after GEAR is applied to baseline KBQA models. However, in SimpleQuestions, the improvement in accuracy on answerable questions is inapparent. This is reasonable because overfitting on negative samples mainly presents in unanswerable questions. This improvement demonstrates that GEAR improves the performance of KBQA particularly on identifying unanswerable questions.

Evaluation of Seq2seq in Generation Module We further discuss the performance of the seq2seq model in Generation Module. The quality of the generated sequence reflects the general text generation and semantic knowledge contained in the seq2seq model. As shown in Table 2, the performance of generated predicate is evaluated by four metrics. EM, BLEU-1 and ROUGE-L are metrics that measure the n-gram overlap between a reference predicate and a candidate predicate, which are not always valid factuality metrics. Metric HCI [5] is a human evaluation method to test whether the predicate generated by GEAR is a reliable predicate from the perspective of humans and how effective the perception of the end-user towards is accepting the "No Answer". Ten volunteers were asked to rate the accuracy of the generated predicates. We select 1000 questions at random from the two datasets. Each of these volunteers is assigned 500 questions to rate the answers given by the KBQA system. We guarantee that each question will be asked five times. For each question, we use hard voting to calculate the final result. The results demonstrate that our seq2seq model is capable of generating a proper predicate in response to a given question.

5 Conclusion

In this paper, we present GEAR, a generation based framework to improve the performance of KBQA task. We also propose PDR, an algorithm to implement the ensemble of generation based model and traditional retrieval based KBQA model. We illustrate that the proposed framework can successfully reduce the noise of negative samples from an incomplete KG. Moreover, the framework improves the ability of a KBQA system to identify unanswerable questions. Empirical results on two popular KBQA datasets under different degrees of KG incompleteness and case study demonstrate the effectiveness of our model.

Acknowledgements We thank anonymous reviewers for their comments and suggestions. This work was supported by National Key Research and Development Project (No.2020AAA0109302), Shanghai Science and Technology Innovation Action Plan (No.19511120400) and Shanghai Municipal Science and Technology Major Project (No.2021SHZDZX0103).

References

1. Bordes, A., Usunier, N., Chopra, S., Weston, J.: Large-scale simple question answering with memory networks. CoRR (2015)
2. Chen, Y., Wu, L., Zaki, M.J.: Bidirectional attentive memory networks for question answering over knowledge bases. In: Proc. of ACL (2019)
3. Dong, L., Wei, F., Zhou, M., Xu, K.: Question answering over Freebase with multi-column convolutional neural networks. In: Proc. of ACL (2015)
4. Duan, N.: Overview of the nlpcc-iccpol 2016 shared task: open domain chinese question answering. In: Natural Language Understanding and Intelligent Applications (2016)
5. Ehsan, U.: On design and evaluation of human-centered explainable ai systems. In: In Human-Centered Machine Learning Perspectives Workshop at CHI (2019)
6. Godin, F., Kumar, A., Mittal, A.: Learning when not to answer: a ternary reward structure for reinforcement learning based question answering. In: Proc. of ACL (2019)
7. Kim, N., Pavlick, E., Karagol Ayan, B., Ramachandran, D.: Which linguist invented the lightbulb? presupposition verification for question-answering. In: Proc. of ACL (2021)
8. Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., Zettlemoyer, L.: BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In: Proc. of ACL (2020)
9. Liu, A., Huang, Z., Lu, H., Wang, X., Yuan, C.: Bb-kbqa: Bert-based knowledge base question answering. In: Proc. of CCL (2019)
10. Petrochuk, M., Zettlemoyer, L.: SimpleQuestions nearly solved: A new upperbound and baseline approach. In: Proc. of EMNLP (2018)
11. Xiong, W., Yu, M., Chang, S., Guo, X., Wang, W.Y.: Improving question answering over incomplete KBs with knowledge-aware reader. In: Proc. of ACL (2019)
12. Zhu, S., Cheng, X., Su, S.: Knowledge-based question answering by tree-to-sequence learning. Neurocomputing (2020)