Logical Form Generation via Multi-task Learning for Complex Question Answering over Knowledge Bases

Xixin Hu, Xuan Wu, Yiheng Shu, Yuzhong Qu
State Key Laboratory for Novel Software Technology, Nanjing University, China
{xixinhu,xuanwu,yhshu}@smail.nju.edu.cn, yzqu@nju.edu.cn

Abstract

Question answering over knowledge bases (KBQA) for complex questions is a challenging task in natural language processing. Recently, generation-based methods that translate natural language questions to executable logical forms have achieved promising performance. These methods use auxiliary information to augment the logical form generation of questions with unseen KB items or novel combinations, but the noise introduced can also leads to more incorrect results. In this work, we propose GMT-KBQA, a **G**eneration-based KBQA method via **M**ulti-**T**ask learning, to better retrieve and utilize auxiliary information. GMT-KBQA first obtains candidate entities and relations through dense retrieval, and then introduces a multi-task model which jointly learns entity disambiguation, relation classification, and logical form generation. Experimental results show that GMT-KBQA achieves state-of-the-art results on both ComplexWebQuestions and WebQuestionsSP datasets. Furthermore, the detailed evaluation demonstrates that GMT-KBQA benefits from the auxiliary tasks and has a strong generalization capability.[1]

1 Introduction

Question answering over knowledge bases (KBQA) is the task of answering natural language questions based on the facts stored in knowledge bases (KBs). In recent years, an increasing number of KBQA methods arise with the emergence of large-scale KBs, such as Freebase (Bollacker et al., 2008), DBpedia (Lehmann et al., 2015) and Wikidata (Vrandečić and Krötzsch, 2014). A mainstream paradigm of KBQA methods is semantic parsing (SP) (Berant et al., 2013), where natural language questions are parsed into logical forms such as λ-DCS (Liang, 2013), SPARQL (Pérez et al., 2009), S-expression (Gu et al., 2021), etc. However, complex questions that involve reasoning over multiple entities, relations, or constraints remain a challenge for SP-based methods. Most of the SP-based methods use a pipeline including entity/relation linking, constraint detection, and logical form building (Singh et al., 2018; Hu et al., 2021). As complex questions require multiple entities and relations, errors introduced by previous linkers reduce the performance of the pipeline.

With the success of applying natural language generation to various tasks (Raffel et al., 2019; Rothe et al., 2020), recent KBQA methods (Cao et al., 2022; Yin et al., 2021; Gu et al., 2021) cast semantic parsing to a logical form generation task in a sequence-to-sequence (Seq2Seq) manner, fine-tuning pre-trained encoder-decoder models to generate logical forms from natural language questions. However, it is impractical for a simple Seq2Seq model to generate unseen entities and relations that never appear in the training set. To alleviate such cases, more generation-based methods (Huang et al., 2021; Das et al., 2021; Ye et al., 2021) augment logical form generation with auxiliary information such as linked entities (Huang et al., 2021), similar question-query pairs (Das et al., 2021), candidate logical forms (Ye et al., 2021), etc. Auxiliary information enhances the generalization capability of the generation models, but can also lead to incorrect results due to the noisy information introduced. We find that a Seq2Seq model can generate exact logical forms for around 92% questions in ComplexWebQuestions dataset if provided with golden entities and relations. However, the proportion drops drastically to 51% if the linking results are from practical linkers. This shows that Seq2Seq models can construct correct logical forms augmented with auxiliary information, and the quality of the auxiliary information has a great impact on the generated results.

Inspired by this discovery, we propose a generation-based KBQA method GMT-KBQA (**G**eneration via **M**ulti-**T**ask learning) that learns to

[1]Our code is available at https://github.com/HXX97/GMT-KBQA.

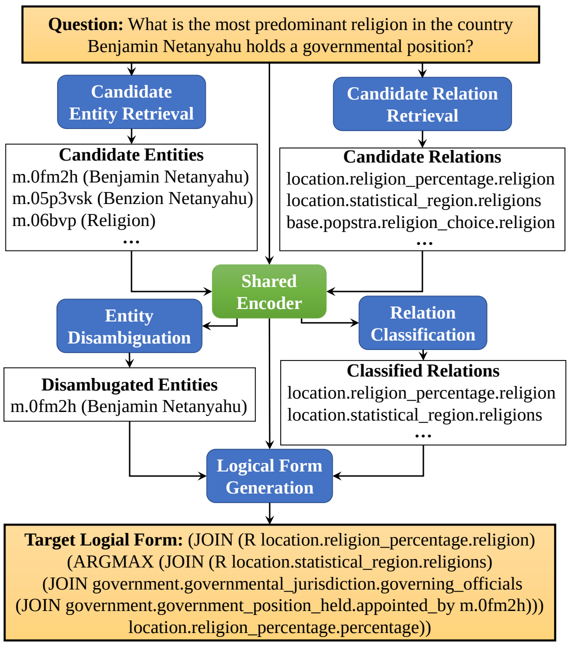

Question: What is the most predominant religion in the country Benjamin Netanyahu holds a governmental position?
Candidate Entity Retrieval
Candidate Relation Retrieval
Candidate Entities
m.0fm2h (Benjamin Netanyahu)
m.05p3vsk (Benzion Netanyahu)
m.06bvp (Religion)
...
Candidate Relations
location.religion_percentage.religion
location.statistical_region.religions
base.popstra.religion_choice.religion
...
Shared Encoder
Entity Disambiguation
Relation Classification
Disambugated Entities
m.0fm2h (Benjamin Netanyahu)
Classified Relations
location.religion_percentage.religion
location.statistical_region.religions
...
Logical Form Generation
Target Logial Form: (JOIN (R location.religion_percentage.religion) (ARGMAX (JOIN (R location.statistical_region.religions) (JOIN government.governmental_jurisdiction.governing_officials (JOIN government.government_position_held.appointed_by m.0fm2h))) location.religion_percentage.percentage))

Figure 1: Overview of GMT-KBQA. The T5 encoder is learned with multiple tasks including the entity/relation classification and the logical form generation.

refine the auxiliary information and generate logical forms at the same time. Figure 1 shows the overview of our proposed method. The core of GMT-KBQA is an encoder-decoder model jointly trained with a translation task for logical form generation and two auxiliary tasks: entity disambiguation and relation classification. Our method attempts to improve logical form generation by sharing the parameters within related tasks. Instead of linking the entities and relations by off-the-shelf linkers before logical form generation, we retrieve candidate entities and relations in a dense space, leaving entity disambiguation and relation classification as auxiliary tasks.

The main contributions of this work are as follows:

1. We propose a generation-based KBQA method via multi-task learning (GMT-KBQA), where the logical form generation task is jointly trained with two auxiliary tasks. GMT-KBQA retrieves auxiliary information including candidate relations and entities through dense retrieval, which balances coverage and efficiency. The refined auxiliary information enables GMT-KBQA to generate unseen KB items.

2. Experimental results demonstrate that our method outperforms previous methods on both WebQuestionsSP and ComplexWebQuestions. Further analysis shows that our method benefits from the multi-task learning framework and achieves better performances on both logical form generation and the two auxiliary tasks.

2 Methodology

This section details the GMT-KBQA method. Given a natural language question, we first retrieve auxiliary information including candidate entities and relations by dense retrieval. Then we refine the retrieved auxiliary information and generate the target logical form via multi-task learning. As shown in Figure 1, the three tasks share a common encoder, and each task has an individual layer on top of the encoder. Details of our method will be given in the following subsections.

2.1 Preliminaries

A knowledge graph is a collection of subject-relation-object triples in the form of (s,r,o), where s is an entity, r is a relation, and o can be entities or literals (e.g., text descriptions, numeric values, date-time, etc).

Given a natural language question, our method aims at generating a logical form that can be executed over the KB. Following Gu et al. (2021) and Ye et al. (2021), we use S-expressions as the target logical forms. Since most KB storage engines only support SPARQL queries, we finally convert S-expressions into equivalent SPARQL queries to get answers following Gu et al. (2021).

2.2 Retrieval of Auxiliary Information

Existing generation-based methods retrieve similar question-query pairs (Das et al., 2021) or enumerate candidate logical forms (Ye et al., 2021) as auxiliary information, but the coverage of cases cannot be guaranteed and the enumeration of logical forms can be time-consuming. Instead, we retrieve candidate entities and relations in a dense space as auxiliary information, which balances coverage and efficiency.

Candidate entity retrieval Most KBQA methods (Yih et al., 2015; Sun et al., 2019; Lan and Jiang, 2020; Huang et al., 2021; Ye et al., 2021) take entity linking as the first step, whereas the result of entity linking determines the upper bound of the final performance. We conduct entity retrieval to get candidate entities with high coverage, deferring entity disambiguation until logical form generation by multi-task learning. Specifically, we use ELQ (Li et al., 2020), an end-to-end entity linking model through dense retrieval. To further improve the coverage of candidate entities, we use a large entity mention map provided by FACC1 project (Gabrilovich et al., 2013) to retrieve entities not linked by ELQ. For each question, we merge top-$k/2$ candidate entities from each linking model, to retain top-k ranked entities. If the number of candidate entities is less than k, entities randomly sampled from the training set will be supplemented.

Candidate relation retrieval Inspired by the zero-shot entity linking work with dense retrieval (Wu et al., 2020), we design a two-stage relation retrieval module utilizing the bi-encoder and cross-encoder architecture, which is shown in Figure 2.

In the first stage, we train a bi-encoder that embeds questions and relations into the same dense space with two independent BERT (Devlin et al., 2019) encoders.

Specifically, inspired by Das et al. (2021), for each question q, we mask entity mentions detected in candidate entity retrieval stage with $[BLANK]$ token. And we denote the question with entity mentions masked as τ_q.

In addition, a relation r is represented as:

$$r \mid \text{label}_r \mid \text{domain}_r \mid \text{range}_r$$

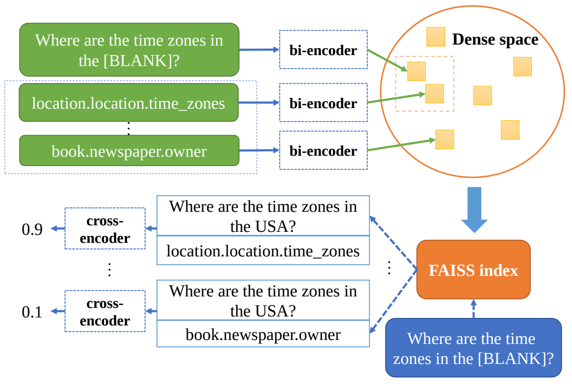
where label_r, domain_r and range_r are the meta descriptions of r in KB. For example, relation *location.location.time_zones* is represented as *location.location.time_zones* | *Time zone(s)* | *location.location* | *time.time_zone*. This enriched form of relation is denoted as τ_r. A question q and a relation r are encoded into vectors:

$$\begin{aligned} \mathbf{y}_q &= \textsc{BertCls}_1(\tau_q) \\ \mathbf{y}_r &= \textsc{BertCls}_2(\tau_r) \end{aligned} \quad (1)$$

where BertCls denotes the [CLS] representation of the input. The relevance score of question q and relation r is computed by dot-product:

$$s_b(q, r) = \mathbf{y}_q \cdot \mathbf{y}_r \quad (2)$$

For each pair of a question and its relevant relation (q, r_i), we randomly sample $B-1$ relations that are not in the logical form of the question to construct a batch consisting of B training pairs. The optimization goal is to maximize the score of the relevant



Where are the time zones in the [BLANK]?
location.location.time_zones
book.newspaper.owner
bi-encoder
bi-encoder
bi-encoder
Dense space
0.9
cross-encoder
Where are the time zones in the USA?
location.location.time_zones
0.1
cross-encoder
Where are the time zones in the USA?
book.newspaper.owner
FAISS index
Where are the time zones in the [BLANK]?

Figure 2: Overview of our two-stage relation retrieval method utilizing the bi-encoder and cross-encoder.

relation against randomly sampled relations, and the loss is computed as:

$$\mathcal{L}(q, r_i) = -s_b(q, r_i) + \log \sum_{j=1}^{B} \exp(s_b(q, r_j)) \quad (3)$$

After training the bi-encoder, relation representations are cached for inference efficiency. For an incoming question, we embed it to a vector by the bi-encoder and then use FAISS (Johnson et al., 2019) to retrieve the nearest relations.

In the second stage, the retrieved relations for a question are re-ranked with a cross-encoder to get the most relevant relations. The cross-encoder is a single BERT model that takes the concatenation of the question and its candidate relation as input. Compared with the bi-encoder, the cross-encoder has deep cross attention between the question and the relation.

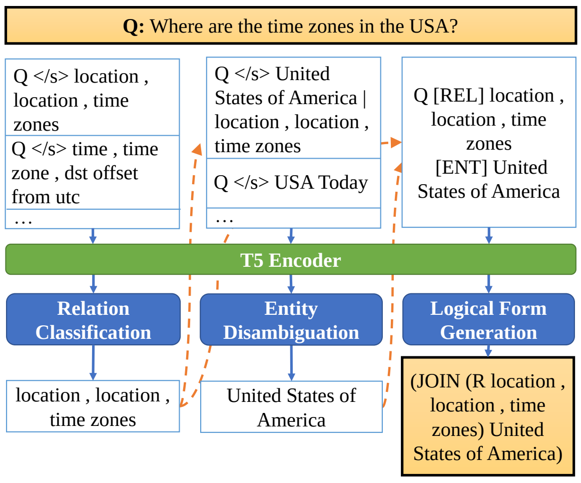
The input of cross-encoder is the concatenation of question q and candidate relation r. The relevance score of q and r is:

$$s_c(q, r) = \textsc{Linear}(\textsc{BertCls}([q; r])) \quad (4)$$

where Linear is a layer that projects the representation to a binary probability distribution. We train cross-encoder using cross-entropy loss. Finally, top-k candidate relations ranked by cross-encoder are retained.

2.3 Logical Form Generation via Multi-task Learning

After the retrieval of auxiliary information including candidate entities and relations, we introduce a multi-task model that learns to refine the auxiliary information and generate the target logical form as



Q: Where are the time zones in the USA?
Q </s> location , location , time zones
Q </s> time , time zone , dst offset from utc
…
Q </s> United States of America | location , location , time zones
Q </s> USA Today
…
Q [REL] location , location , time zones [ENT] United States of America
T5 Encoder
Relation Classification
Entity Disambiguation
Logical Form Generation
location , location , time zones
United States of America
(JOIN (R location , location , time zones) United States of America)

Figure 3: Multi-task model for logical form generation where "Q" is short for the original question. All tasks share a T5 encoder to encode inputs, and the results of prior stages will be leveraged in the latter stages as the dotted lines indicate.

Figure 3 shows. The backbone of our model is T5 (Raffel et al., 2019), an encoder-decoder structured Seq2Seq model that achieves strong performances on several generation tasks. We use a shared T5 encoder to obtain the representations of the inputs, and the representations are fed to individual networks for different tasks. We define two auxiliary tasks: relation classification and entity disambiguation to enhance the logical form generation task.

Relation classification Given a question q and its retrieved candidate relations R, the relation classification task aims to select the correct relations from R that compose the target logical form. We cast this task to a sentence-pair classification task. The concatenation of question q and relation r is fed to a T5 encoder, and it is represented by the average pooling of the encoder output:

$$\mathbf{y_{q,r}} = \textsc{AvgPool}(\textsc{T5Encoder}([q;r])) \quad (5)$$

Then the representation is projected to a scalar score through a linear projection, and then activated by a sigmoid function:

$$s(q,r) = \textsc{Sigmoid}(\textsc{Linear}(\mathbf{y}_{q,r})) \quad (6)$$

We use binary cross-entropy loss for the relation classification task:

$$\mathcal{L}^{\text{REL}} = -\frac{1}{k}\sum_{i=1}^{k}[u_i \cdot \log(s(q,r_i)) + (1-u_i)\cdot \log(1-s(q,r_i))] \quad (7)$$

where u_i denotes the classification label of relation r_i, and k is the number of candidate relations. The relations classified as positive are denoted as R_q.

Entity disambiguation The retrieved candidate entities usually contain ambiguity, where multiple KB entities are retrieved for one mention. To select the exact entities in the question, we cast entity disambiguation as a sentence-pair classification task similar to the relation classification task. Following Ye et al. (2021), we leverage adjacent relations to help determine if an entity should be linked by a mention. Specifically, we concatenate the label of entity e, with its adjacent KB relations in the form of:

$$\text{label}_e \ | \ r_1 \ | \ r_2 \ | \ r_3 \ | \ \ldots$$

This rich form of entity e is denoted as τ_e. Note that we only concatenate relations classified as relevant to the question by the relation classification task. The concatenation of question q with τ_e is fed to the shared T5 encoder, and the output is averagely pooled:

$$\mathbf{y}_{q,e} = \textsc{AvgPool}(\textsc{T5Encoder}([q;\tau_e])) \quad (8)$$

The relevance score of q and e is computed as:

$$s(q,e) = \textsc{Sigmoid}(\textsc{Linear}(\mathbf{y}_{q,e})) \quad (9)$$

Similarly, we use binary cross-entropy loss for the entity disambiguation task:

$$\mathcal{L}^{\text{ENT}} = -\frac{1}{k}\sum_{i=1}^{k}[v_i \cdot \log(s(q,e_i)) + (1-v_i)\cdot \log(1-s(q,e_i))] \quad (10)$$

where v_i is the classification label of entity e_i, and k is the number of candidate entities. The entities classified as positive are denoted as E_q.

Logical form generation The task of logical form generation is to generate the target logical form of question q given disambiguated entities E_q and classified relations R_q. Following previous generation-based methods (Das et al., 2021; Ye et al., 2021), we construct the inputs by concatenating the question q with relations in R_q and entities in E_q in the form as:

$$q \ \texttt{[REL]} \ r_1 \ \texttt{[REL]} \ r_2, \ \ldots \ \texttt{[ENT]} \ \text{label}_{e_1} \ \texttt{[ENT]} \ \text{label}_{e_2}, \ldots$$

This concatenation is denoted as τ_{all} and fed to the T5 encoder shared with the above-mentioned two auxiliary tasks to obtain the representations:

$$[\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n] = \text{T5E}\textsc{ncoder}(\tau_{all}) \tag{11}$$

where $\mathbf{h}_i$ denotes the representation of the i-th token of the input and n is the number of tokens in τ_{all}. Then we use a T5 decoder to decode the representations into a logical form token by token. Assuming that the target logical form consists of m tokens $a_1, \ldots, a_m$, we calculate cross-entropy loss with teacher forcing:

$$\begin{aligned} \mathbf{p}_j &= \text{T5D}\textsc{ecoder}(a_1, \ldots, a_{j-1}, \mathbf{h}_1, \ldots, \mathbf{h}_n) \\ \mathcal{L}^{\text{GEN}} &= -\frac{1}{m}\sum_{j=1}^{m} \log \mathbf{p}_{j,a_j} \end{aligned} \tag{12}$$

where $\mathbf{p}_j$ denotes the probability distribution over the decoding vocabulary at the j-th step, and $\mathbf{p}_{j,a_j}$ represents the probability of token a_j.

Training Objective We jointly train the relation classifier, entity disambiguator, and logical form generator with a combined loss:

$$\mathcal{L} = \mathcal{L}^{\text{REL}} + \mathcal{L}^{\text{ENT}} + \mathcal{L}^{\text{GEN}} \tag{13}$$

This training objective enables the generator to learn from auxiliary tasks, where the target logical form not only supervises the generation task but also supervises entity disambiguation and relation classification.

3 Evaluation

3.1 Setups

Datasets All the experiments are conducted on WEBQUESTIONSSP (WebQSP) (Yih et al., 2016) and COMPLEXWEBQUESTIONS (CWQ) (Talmor and Berant, 2018) datasets. Both datasets are based on Freebase (Bollacker et al., 2008).

WebQSP consists of 4,737 questions labeled with SPARQL queries. Most questions of WebQSP require up to 2 hops of reasoning.

CWQ contains 34,689 questions with SPARQL queries. These questions are obtained by extending the questions in WebQSP to increase the complexity. Questions in CWQ may require up to 4-hops reasoning, making it quite challenging.

Hyperparameters We use T5-base and BERT-base-uncased implementation from HuggingFace[2]. The sample size B for training the bi-encoder in relation retrieval (Section 2.2) is set to 100, and the top 100 nearest relations are searched by the FAISS index. The number of candidate k is set to 10 for both entity and relation retrieval. Beam search is utilized in the decoding process, and we set beam size to 50 by default. For CWQ, our multi-task models are trained for 15 epochs, with training batch size set to 8 and inference batch size set to 4 due to GPU memory limits. For WebQSP, models are trained for 20 epochs with batch size set to 2 due to less data volume.

Implementation details Since neither WebQSP nor CWQ provides golden S-expressions, we follow the implementation of Ye et al. (2021) to convert a golden SPARQL query to its equivalent S-expression. WebQSP provides more than one SPARQL annotation for some questions, and we choose the shortest SPARQL query that can be successfully converted to S-expression.

The generation target of our model is normalized S-expression, where KB relations are split into tokens and entities are represented with their labels. Thus, a post-process step is needed to convert generated normalized S-expression to its original form. Specifically, entity labels are mapped into entity ids in the KB based on the output of the entity disambiguation task; normalized relations are converted back based on rules. Finally, S-expression is converted to SPARQL to be executed against KB in the same way of Gu et al. (2021).

In the training phase, the results of auxiliary tasks are not steady at the beginning, since the model parameters are not well-trained. Therefore, we do not concatenate the output of prior tasks as described in Section 2.3 for the first 5 training epochs.

In the inference phase, we utilize KB to validate generated logical forms. Given a question, we generate a bunch of logical forms by beam search and they are executed in turn until a non-empty query result is returned. It helps to filter invalid logical forms.

Metrics Following Das et al. (2021), we use the standard evaluation metrics, namely precision (P), recall (R), macro F1 (F1), and accuracy (Acc.).

[2]https://huggingface.co/

	CWQ		WebQSP	
Method	**Acc.**	**F1**	**Acc.**	**F1**
QGG (Lan and Jiang, 2020)	-	40.4	-	74.0
BART-large (Huang et al., 2021)	-	68.2	-	74.6
CBR-KBQA (Das et al., 2021)	67.1	70.0	69.9	72.8
ReTraCk* (Chen et al., 2021)	-	-	-	74.7
RnG-KBQA (Ye et al., 2021)	-	-	71.1	75.6
GMT-KBQA (Ours)	**72.2**	**77.0**	**73.1**	**76.6**

Table 1: QA evaluation results (%) on CWQ test set and WebQSP test set. * denotes using oracle entity linking results. Acc.: Accuracy.

	CWQ						WebQSP					
	Entity linking			Relation linking			Entity linking			Relation linking		
Methods	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
Retriever	71.6	**70.5**	68.8	84.9	74.2	77.3	53.9	**86.0**	62.6	66.3	68.7	65.4
GMT-KBQA	**78.8**	69.4	**72.3**	**85.2**	**81.0**	**81.4**	**79.9**	75.3	**76.1**	**74.1**	**73.9**	**72.1**

Table 2: Entity and relation linking evaluation (%) of our retriever and multi-task model.

3.2 Experimental Results

QA performance Table 1 summarizes evaluation results on both CWQ and WebQSP dataset. The results of other methods are taken from corresponding papers directly. The result shows that our method sets the new state-of-the-art on CWQ dataset by a large margin, surpassing CBR-KBQA (Das et al., 2021) by 5.1% accuracy and 7.0% F1. CBR-KBQA retrieves similar question-query pairs in the training set to augment logical form generation and introduces a revision step for relations not covered by the cases, whereas our method retrieves relevant entities and relations from the entire KB through dense retrieval, which balances coverage and efficiency. Huang et al. (2021) utilize auxiliary information including candidate entities and generate entity text labels instead of entity IDs. Our method achieves a notable advantage against Huang et al. (2021), indicating that our method makes better use of auxiliary information and is more capable of dealing with unseen KB items.

GMT-KBQA also achieves new state-of-the-art results on WebQSP dataset, where the questions are relatively simpler than CWQ. According to Table 1, our method outperforms existing methods even if they use oracle entity annotations (Chen et al., 2021). RnG-KBQA (Ye et al., 2021), the previous state-of-the-art on WebQSP, combines ranking and generation for both coverage and generalization. Our method obtains an increase of 2.0% accuracy and 1.0% F1 against RnG-KBQA although we only rely on the generation results. In summary, the results on CWQ and WebQSP datasets suggest that our method is effective in solving questions with different complexity.

Improvement on auxiliary tasks GMT-KBQA consists of three tasks, namely entity disambiguation, relation classification, and logical form generation. Apart from the experiments on QA tasks, also we evaluate the entity and relation linking results to show the impact of multi-task learning.

As shown in Table 2, for Retriever, candidate entities/relations retrieved in Section 2.2 are disambiguated with prediction scores in retrieval stage. For GMT-KBQA, candidate entities/relations are disambiguated with the output of entity disambiguation task and relation classification task described in Section 2.3 respectively. Experiment result indicates the improvement of our model on both entity linking and relation linking performance. The entity linking F1 is improved by 3.5% and 13.5% on CWQ and WebQSP, respectively. Although our disambiguation model sacrifices recall to some extent, we can observe a significant increase in precision. The experiment results further prove the effectiveness of our disambiguation ap-

	CWQ		WebQSP	
Methods	**F1**	Δ	**F1**	Δ
GMT-KBQA	**77.0**	-	**76.6**	-
w/o Entity	72.7	-4.3	74.9	-1.7
w/o Relation	75.9	-1.1	75.1	-1.5
w/o Entity, Relation	74.0	-3.0	74.7	-1.9

Table 3: QA performance (%) of variants of our model. *Entity* and *Relation* are short for entity disambiguation task and relation classification task respectively.

	CWQ		WebQSP	
Methods	**F1**	Δ	**F1**	Δ
T5-base	74.0	-	74.7	-
w/ Retrieval	71.4	- 2.6	72.9	- 1.8
w/ Oracle	**94.1**	+20.1	**94.8**	+20.1
GMT-KBQA	77.0	+ 3.0	76.6	+ 1.9

Table 4: Impact of auxiliary information on the generation model. *Retrieval*/*Oracle* are short for concatenating retrieved/oracle auxiliary information to input.

proach: the entity representation can be enriched with retrieved relations, which provides sufficient evidence for the disambiguation model. The relation linking F1 is also improved by 4.1% and 6.7% on two datasets, which proves the effectiveness of our multi-task model for relational linking.

3.3 Analysis

Ablation study To further illustrate the impact of each task and their combination, we evaluate QA results with different model variants, i.e., with task(s) removed. Table 3 shows 1) our final model substantially outperforms other model variants, indicating that our multi-task setting properly organizes and makes full use of all the tasks; 2) performance drops in general with more tasks removed, which shows the necessity of the auxiliary tasks.

Impact of auxiliary information Table 4 shows that a giant improvement can be achieved with oracle entities and relations, which confirms the importance and potential of utilizing auxiliary information.

However, for practical scenarios without the oracle, directly concatenating linking results leads to a decrease in performance compared to the vanilla generation model (T5-base in Table 4). The results are consistent with our motivation that auxiliary information could be better utilized via a multi-task setting instead of simple concatenation.

	CWQ			WebQSP		
Methods	**P**	**R**	**F1**	**P**	**R**	**F1**
T5-base	64.8	67.0	64.9	64.3	68.5	64.5
GMT-KBQA	**68.3**	**70.9**	**68.5**	**66.7**	**69.8**	**66.6**

Table 5: QA results (%) on test set questions with unseen KB items.

There are two advantages of GMT-KBQA to achieve the improvement. First, GMT-KBQA jointly trains the three tasks with a combined loss, improving the accuracy of each task and resulting in better overall performance. Second, when noisy candidate relations/entities are provided, for the model with simple concatenation, although given a generation loss, there's no explicit clue for the model to locate the noisy candidates, which leads to confusion in the optimization stage. As for GMT-KBQA with a multi-task setting, the loss of relation classification and entity disambiguation helps the model locate mis-classified candidates, making the optimization objective clearer. In this way, candidate relations/entities with higher precision and recall are given to the model, making it easier to generate correct logical forms.

Performance on unseen KB items To measure our model's capability of handling unseen relations and entities, experiments are conducted on questions from CWQ/WebQSP test set whose golden SPARQL contains unseen entities or relations (compared to the training set). Evaluation result in Table 5 indicates that GMT-KBQA contributes to 3.6% and 2.1% F1 increase in CWQ and WebQSP respectively. This experiment demonstrates that our method leads to improvement in generalization capability compared to the vanilla generation model.

Effect of beam size To study the impact of different beam sizes on the decoding stage, the QA performance of GMT-KBQA with different beam sizes is evaluated as shown in Table 6, where we also list the performance of other methods utilizing beam search. As the result implies, performance improves with a larger beam size, which indicates that beam search combined with the execution checking (Section 3.1) helps to discover valid and correct logical forms. The experiment further proves that our model still achieves great

	CWQ					WebQSP				
Methods	1	5	10	50	100	1	5	10	50	100
CBR-KBQA (Das et al., 2021)	-	70.0	-	-	-	-	72.8	-	-	-
BART-large (Huang et al., 2021)	55.5	-	60.9	-	68.2	67.5	-	73.6	-	74.6
GMT-KBQA(Ours)	**62.2**	**72.7**	**74.4**	**77.0**	-	**70.5**	**75.2**	**76.5**	**76.6**	-

Table 6: F1 metrics (%) with different beam sizes.

performance with smaller beam sizes, outperforming other methods with the same beam size.

Error analysis We analyze the questions not answered correctly by GMT-KBQA in the CWQ test set. The errors can be summarized as follows.

- **Structure generation error** (54.5%). An example of this error is the wrong choice of function. For instance, in question *"What is the youngest college that Harry S Truman attend?"*, GMT-KBQA fails to understand that "youngest" indicates the shortest existence time of a college, i.e., the latest foundation time. Therefore, our model applies "AGRMIN" operation instead of the correct "AGRMAX" function on the foundation time.

- **Relation linking error** (16.4%) and **entity linking error** (12.3%). Despite the efforts we put into relation classification and entity disambiguation, linking errors cannot be completely avoided. For example. in question *"What form of currency was used in the place where Nicolas Sarkozy was governor before the Euro was established?"*, linking correct relation *"location.country.currency_formerly_used"* requires understanding of past tense, i.e., *"was used"* in the question.

- **S-expression conversion** (11.1%) and **De-normalization** (4.6%). Some overly complex SPARQL queries do not have equivalent S-expression or the execution result of converted S-expression differs from the original SPARQL query. Apart from that, our de-normalization phase also causes some errors.

4 Related Work

Existing KBQA methods can be mainly divided into two categories: information retrieval-based methods (IR-based methods) and semantic parsing-based methods (SP-based methods).

IR-based methods (Bordes et al., 2015; Dong et al., 2015; Hao et al., 2017; Zhao et al., 2019) follow a retrieval-and-rank paradigm. They first retrieve a question-specific graph from the KB and then rank entities in the graph by their relevance to the question. For complex questions, recent IR-based methods turn their attention to graph retrieval (Sun et al., 2019; Saxena et al., 2020) and multi-hop reasoning over graphs (Zhou et al., 2018; He et al., 2021; Shi et al., 2021). Generally, IR-based methods fit into end-to-end training, but they lack interpretability because of the black-box reasoning process.

SP-based methods, closely relevant to our method, are more transparent compared with IR-based methods. They answer questions by parsing them into logical forms executable against KBs, including λ-DCS (Berant et al., 2013), SPARQL (Huang et al., 2021; Das et al., 2021), query graph (Yih et al., 2015; Bao et al., 2016; Lan and Jiang, 2020), and S-expression (Gu et al., 2021; Ye et al., 2021). Past SP-based methods parse questions with a bottom-up semantic parser (Berant et al., 2013) or iteratively generate and rank candidate query graphs (Yih et al., 2015; Lan and Jiang, 2020). However, previous semantic parsers have limited coverage for diverse complex queries (Lan and Jiang, 2020), and query graph generation methods suffer from the high computational cost of expanding the graphs (Qin et al., 2021). Recent methods (Zhang et al., 2019; Yin et al., 2021; Huang et al., 2021) take advantage of language generation models to directly generate executable logical forms from questions. As vanilla generation models do not generalize well to questions on KB items with novel combinations and unseen ones, Das et al. (2021) generate complex logical forms conditioned on retrieved similar questions along with their logical forms, but they need to add human-labeled cases to cover absent relations in the case memory. Ye et al. (2021) first rank a pool of candidate logical forms obtained by enumerating relation paths over the KBs, and then generate the final logical form based on the question combined with top-ranked candidates. The enumeration of candidate logical forms involves a large search space on the KB, which is time-consuming and computationally expensive.

5 Conclusion

We present GMT-KBQA to improve the generalization capability of generation-based methods utilizing auxiliary information. GMT-KBQA first retrieves candidate entities and relations in dense space. Then, its multi-task learning framework learns to refine the auxiliary information along with generating target logical forms at the same time. Experimental results on two datasets, CWQ and WebQSP, show that our method sets new state-of-the-art by a large margin. The further analysis illustrates that our logical form generation task and auxiliary tasks benefit from each other and GMT-KBQA achieves strong performance for unseen KB items. In general, GMT-KBQA gives an insight into generating more accurate logical forms through auxiliary information retrieval and multi-task learning. Currently, answering questions that require in-depth understanding such as logical or commonsense reasoning remains a challenging task, and we will strengthen GMT-KBQA for such scenarios in the future.

Acknowledgements

This work was supported in part by the NSFC (62072224). The authors would like to thank all the participants of this work and anonymous reviewers.

References

Junwei Bao, Nan Duan, Zhao Yan, Ming Zhou, and Tiejun Zhao. 2016. Constraint-based question answering with knowledge graph. In *Proceedings of COLING 2016*, pages 2503–2514.

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

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the SIGMOD 2008*, pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

Shulin Cao, Jiaxin Shi, Liangming Pan, Lunyiu Nie, Yutong Xiang, Lei Hou, Juanzi Li, Bin He, and Hanwang Zhang. 2022. KQA Pro: A large diagnostic dataset for complex question answering over knowledge base. In *Proceedings of the ACL 2022*.

Shuang Chen, Qian Liu, Zhiwei Yu, Chin-Yew Lin, Jian-Guang Lou, and Feng Jiang. 2021. Retrack: A flexible and efficient framework for knowledge base question answering. In *Proceedings of the ACL-IJCNLP 2021*, pages 325–336.

Rajarshi Das, Manzil Zaheer, Dung Thai, Ameya Godbole, Ethan Perez, Jay Yoon Lee, Lizhen Tan, Lazaros Polymenakos, and Andrew McCallum. 2021. Case-based reasoning for natural language queries over knowledge bases. In *Proceedings of the EMNLP 2021*, pages 9594–9611.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the NAACL-HLT 2019*, pages 4171–4186. Association for Computational Linguistics.

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

E. Gabrilovich, M. Ringgaard, and A. Subramanya. 2013. Facc1: Freebase annotation of clueweb corpora, version 1 (release date 2013-06-26, format version 1, correction level 0).

Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond iid: three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*, pages 3477–3488.

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

Gaole He, Yunshi Lan, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Improving multi-hop knowledge base question answering by learning intermediate supervision signals. In *Proceedings of the WSDM 2021*, pages 553–561.

Xixin Hu, Yiheng Shu, Xiang Huang, and Yuzhong Qu. 2021. EDG-based question decomposition for complex question answering over knowledge bases. In *Proceedings of the ISWC 2021*, volume 12922 of *Lecture Notes in Computer Science*, pages 128–145. Springer.

Xin Huang, Jung-jae Kim, and Bowei Zou. 2021. Unseen entity handling in complex question answering over knowledge base via language generation. In *Findings of EMNLP 2021*, pages 547–557.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547.

Yunshi Lan and Jing Jiang. 2020. Query graph generation for answering multi-hop complex questions from knowledge bases. In *Proceedings of ACL 2020*. Association for Computational Linguistics.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2015. Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195.

Belinda Z Li, Sewon Min, Srinivasan Iyer, Yashar Mehdad, and Wen-tau Yih. 2020. Efficient one-pass end-to-end entity linking for questions. In *Proceedings of the EMNLP 2020*, pages 6433–6441.

Percy Liang. 2013. Lambda dependency-based compositional semantics. *arXiv preprint arXiv:1309.4408*.

Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. 2009. Semantics and complexity of sparql. *ACM TODS*, 34(3):1–45.

Kechen Qin, Cheng Li, Virgil Pavlu, and Javed Aslam. 2021. Improving query graph generation for complex question answering over knowledge base. In *Proceedings of the EMNLP 2021*, pages 4201–4207.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging pre-trained checkpoints for sequence generation tasks. *TACL*, 8:264–280.

Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of the ACL 2020*, pages 4498–4507.

Jiaxin Shi, Shulin Cao, Lei Hou, Juanzi Li, and Hanwang Zhang. 2021. Transfernet: An effective and transparent framework for multi-hop question answering over relation graph. In *Proceedings of the EMNLP 2021*, pages 4149–4158. Association for Computational Linguistics.

Kuldeep Singh, Arun Sethupat Radhakrishna, Andreas Both, Saeedeh Shekarpour, Ioanna Lytra, Ricardo Usbeck, Akhilesh Vyas, Akmal Khikmatullaev, Dharmen Punjani, Christoph Lange, et al. 2018. Why reinvent the wheel: Let's build question answering systems together. In *Proceedings of the WWW 2018*, pages 1247–1256.

Haitian Sun, Tania Bedrax-Weiss, and William Cohen. 2019. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text. In *Proceedings of the EMNLP-IJCNLP 2019*, pages 2380–2390.

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

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Scalable zero-shot entity linking with dense entity retrieval. In *Proceedings of the EMNLP 2020*, pages 6397–6407. Association for Computational Linguistics.

Xi Ye, Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, and Caiming Xiong. 2021. Rng-kbqa: Generation augmented iterative ranking for knowledge base question answering. In *Proceedings of the ACL 2021*.

Scott Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the ACL-IJCNLP 2015*.

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

Xiaoyu Yin, Dagmar Gromann, and Sebastian Rudolph. 2021. Neural machine translating from natural language to sparql. *Future Generation Computer Systems*, 117:510–519.

Haoyu Zhang, Jingjing Cai, Jianjun Xu, and Ji Wang. 2019. Complex question decomposition for semantic parsing. In *Proceedings of the ACL 2019*, pages 4477–4486.

Wenbo Zhao, Tagyoung Chung, Anuj Goyal, and Angeliki Metallinou. 2019. Simple question answering with subgraph ranking and joint-scoring. In *Proceedings of NAACL-HLT*, pages 324–334.

Mantong Zhou, Minlie Huang, and Xiaoyan Zhu. 2018. An interpretable reasoning network for multi-relation question answering. In *Proceedings of the COLING 2018*, pages 2010–2022.