Unanswerable Question Correction in Question Answering over Personal Knowledge Base

An-Zi Yen,[1] Hen-Hsen Huang,[2,3] Hsin-Hsi Chen[1,3]
[1]Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan
[2]Department of Computer Science, National Chengchi University, Taipei, Taiwan
[3]MOST Joint Research Center for AI Technology and All Vista Healthcare, Taiwan
azyen@nlg.csie.ntu.edu.tw, hhhuang@nccu.edu.tw, hhchen@ntu.edu.tw

Abstract

People often encounter situations where they need to recall past experiences from their daily life. In this paper, we aim to construct a question answering system that enables human to query their past experiences over personal knowledge base. Previous works on knowledge base question answering focus on finding answers for answerable questions. In the real world applications, however, people often muddle up facts and ask those questions that cannot be answered with knowledge base. This work presents a novel system consisting of question answering model and question generation model. It not only answers answerable questions, but also corrects unanswerable questions if necessary. Our question answering model recognizes the question that is inconsistent with the state of the personal knowledge base and suggests facts that can form a feasible question. Then, the facts are converted to an answerable question by the question generation model. For refining question, we propose a question generation model based on the reinforcement learning (RL) with question editing mechanism. Experimental results show that our proposed system is effective for correcting unanswerable questions in personal knowledge base question answering.

Introduction

Knowledge base question answering (KBQA) is a task that takes a natural language question as input and returns a factual answer using structured knowledge bases such as Freebase (Bollacker et al. 2008) and DBpedia (Auer et al. 2007). Previous researches in KBQA (Bordes et al. 2015; Bao et al. 2016; Iyyer, Yih, and Chang 2017; Sorokin and Gurevych 2018; Luo et al. 2018) focus on retrieving information over knowledge base (KB) about world knowledge. An application of human living support that enables people to retrieve personal knowledge, such as daily life experiences and the facts about themselves, remains to be explored.

Question answering over personal knowledge base (PKBQA) is an emerging research topic that provides support for people who may have difficulty recalling past experiences. People often forget something over time, and encounter the situations where they need to recall past experiences in their daily life. Forgetting the names of exact entities from their life is common. Memex (Bush 1945), a concept of lifelogging introduced by Vannevar Bush in 1945, is a hypothetical system for organizing the knowledge of a person in her/his life time. The system is aimed at serving as memory assistance to an individual for querying her/his lifelog in a flexible and effective manner. In our previous work (Yen, Huang, and Chen 2019a,b), we propose a multimodal joint learning approach to construct a personal KB of an individual by extracting personal life events from social media posts. Based on our public personal KB,[1] we address the topic of constructing a system that allows a user to query past experiences in natural language over personal KB. For instance, a user forgets the name of the place he visited last year. The service tries to identify the name of the place by searching the KB about the individual's life events.

In a real application scenario, however, people might ask questions that are unanswerable since people cannot remember each piece of information in her/his past experiences and form a correct question. The reasons why questions cannot be answered over KBs may include (1) KB is incomplete to cover all facts required by the questions (Xiong et al. 2019), (2) the user-generated questions are ill-formed, such as entities or predicates are missing, and contain ungrammatical phrases (Christmann et al. 2019), (3) questions are ambiguous and have more than one interpretation, since different entities may be mapped to the same surface form, and a predicate expressed in natural language may be mapped to various relations (Yahya et al. 2016; Xu et al. 2019), and (4) entities and/or relations mentioned in the question are inconsistent with the facts in the KB. In this paper, we focus on tackling the unanswerable questions caused by the last reason, i.e., the events expressed in the question inconsistent with the facts in the personal KB.

The issue of unanswerable questions has attracted attention in recent years. Rajpurkar et al. (2018) release a machine reading comprehension dataset named SQuAD 2.0, which contains both answerable and unanswerable questions. In SQuAD 2.0, the answer of a question is a span in the given paragraph. The unanswerable questions are defined as a scenario where the answer is not a part of the context. Therefore, the model has no way to select a text span to the question. Different from SQuAD 2.0, the questions in this work require reasoning across multiple facts in the KB, instead of finding the answer span from sentences in a sin-

[1]http://nlg.csie.ntu.edu.tw/nlpresource/LiveKB/

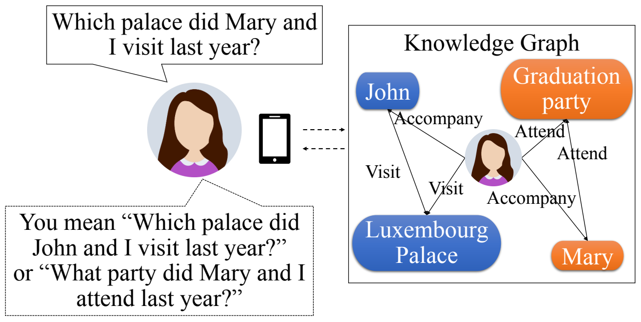
Which palace did Mary and I visit last year?
You mean "Which palace did John and I visit last year?" or "What party did Mary and I attend last year?"
Knowledge Graph
John
Graduation party
Accompany
Attend
Attend
Visit
Visit
Accompany
Luxembourg Palace
Mary

Figure 1: Scenario of PKBQAC for Unanswerable Question.

gle paragraph. In addition, rather than returning no answers to users, it is much better to recommend users the possible questions they might want to ask.

We address the issue of unanswerable questions in PKBQA and aim to suggest users corrected questions. We construct an active PKBQA system that not only retrieves answers from a personal KB for answerable questions, but also suggests possible corrections for unanswerable questions. We refer to such a kind of systems as personal knowledge base question answering with correction (PKBQAC) hereafter. Figure 1 shows a scenario of the use of the PKBQAC system when the question is unanswerable. A user attempts to recall the name of the palace she visited with her friend last year. However, she mixed up with friends' name. That leads to a question that does not match any events stored in her own personal KB. Our system corrects unanswerable questions to help the user recall experiences based on the personal KB.

Specifically, our PKBQAC system consists of a question answering model and a question generation model. The question answering model is aimed at extracting answers from personal KB by generating n candidate query graphs $\{g_1, g_2, ..., g_n\}$. If none of g_i $(1 \leq i \leq n)$ is a sub-graph of the knowledge graph, i.e., the question is unanswerable by the candidate query graphs, the question generation model will be triggered to generate a question according to a corrected query graph g' and the original question.

There are two main challenge issues in this work: (1) how users express questions to recall their past experiences, and (2) how to evaluate if the corrected questions reflect users' intents. Because the recall questions and the corrections for unanswerable questions are hard to collect, we construct a question answering dataset by utilizing a public available personal KB released by Yen et al. (2019a; 2019b). We manually label question answering pairs to simulate the scenario where people query their experiences. The dataset covers several question types, and contains unanswerable questions and their possible corrections. The details will be described in the dataset construction section.

The contributions of this paper are threefold. (1) This work introduces a novel task that extends the detection of unanswerable questions to the correction based on personal KB. (2) We construct a dataset for unanswerable question correction over personal KB. (3) We propose a reinforcement learning (RL) based question generation model with editing mechanism for this task. Experimental results show promising performances are achieved.

Related Work

In this work, we aim to construct a system that can answer user's questions about her/his past experiences over personal KB. The core task can be regarded as general KBQA. Most works in KBQA focus on extracting answers from world KB. Recently, several researches try to develop systems for people to recall the past memory. Jiang et al. (2017) construct a MemexQA dataset, which consists of questions about real-world personal photo albums, and propose a multimodal end-to-end neural network model with an attention kernel for answering questions. Gurrin et al. (2016; 2017; 2019) and Dang-Nguyen et al. (2019) introduce an evaluation dataset for lifelog retrieval systems dealing with querying specific moments in a lifelogger's life. Instead of focusing on retrieving life events from visual data, we construct a question answering dataset based on personal KB.

The previous works on question answering over world KB or lifelogging retrieval focus on answerable questions. In the real world application, people may ask unanswerable questions when recall their experiences. Some researches have addressed the issue of unanswerable questions in machine reading comprehension. Hu et al. (2019) propose an approach to detect unanswerable questions by computing no-answer loss in the objective function. Sun et al. (2018) propose a multi-task learning model to identify the unanswerability of a question. In contrast to answer the right questions only, our system aims at answering questions over KB correctly and actively generate corrected questions related to what the user wants to ask based on her/his personal KB.

Some works improve the performance of KBQA by rewriting or reformulating the input queries. Dong et al. (2017) propose a method to rewrite the input question for increasing the probability of yielding the correct answer. Yahya et al. (2016) propose a framework to relax the user queries by rewriting for achieving higher recall. Xiong et al. (2019) reformulate the query by utilizing the input question representation and the KB knowledge. Buck et al. (2018) propose a RL based system to reformulate the original question for retrieving more candidate answers. These works aim to enhance the ability of retrieving the correct answers by incorporating query expansion mechanism. However, expanding or relaxing unanswerable questions is not suitable to our work, since events described in the original questions are inconsistent with the personal KB. In this paper, we correct the unanswerable questions to answerable ones based on the personal KB.

The difference between our work and the tasks of question generation (Nema et al. 2019; Elsahar, Gravier, and Laforest 2018) and sequence editing (Zhao et al. 2019; Malmi et al. 2019) is that we focus on generating grammatical and answerable questions based on the facts in the KB. Many question generation models have the issues of generating syntactically incorrect or incomplete questions. Seq2seq models (Nema et al. 2019; Chen, Wu, and Zaki 2019) augmented with external reward scores using REINFORCE with the

Combination	Event A	Event B	Event C	Wh-question	Question
EaEb	(User, Ingestion, Espresso, *ts*)	(User, Presence, Eiffel Tower, *ts*)	N/A	what	What did I drink when I went to the Eiffel Tower?
EaEbEc	(User, Presence, Eiffel Tower, *ts*)	(User, Ingestion, Espresso, *ts*)	(User, Presence, Champ de Mars, *ts*)	where	Where did I go after I went to the Champ de Mars for coffee?

Table 1: Examples of the combinations of complex questions. (*ts* denotes timestamp.)

baseline algorithm can generate more valid questions. Inspired by this idea, we propose a question generation model based on the RL with question editing.

Dataset Construction

We utilize our publicly available personal KB to construct the dataset.[2] The personal KB consists of 137 relations and 15,525 events written in Chinese collected from 18 users. In the construction of the dataset, we refer to the annotation workflow by Dubey et al. (2019). We first generate a seed question set by applying several hand-crafted templates based on the events from the personal KB. The hand-crafted templates generate seven types of questions, including "who", "what", "where", "when", "how many", "duration", and "True or False". For each type of the questions, we construct 4 templates. Thus, we have 28 templates in total. We also allow human annotators to write questions on their own if the generated question is ungrammatical or influential. In this way, the generated questions are checked and paraphrased by human annotators to ensure the questions are grammatical. Note that an entity may have multiple surface forms, and KB relations may be expressed in various natural language forms. In addition to the paraphrasing for grammaticality, we also ask human annotators to paraphrase the subjects, the objects, and the predicates in the questions for avoiding exactly matching the entities and relation descriptions in the personal KB.

The types of questions can be further classified into simple questions and complex questions. The simple question (Bordes et al. 2015) means the question can be answered based on only one single relation in the KB. In contrast, complex question (Bao et al. 2016) can be answered by searching from an entity to other entities via two or three relations in the KB. There are two types of complex questions, *EaEb* and *EaEbEc*, where symbols *Ea*, *Eb*, and *Ec* denote different events in the KB. For example, *EaEbEc* means the question is answered by using three events in the personal KB. The answer must be subject, object, or time in event *Ea*. Table 1 shows examples of the two combinations generated from the personal KB.

The true or false questions can be generated by the rules described above. Taking the first row in Table 1 as an example, the true question is "Have I ever been to the Eiffel Tower for Espresso?" The false question can be generated by replacing one of the events. For instance, we replace (User, Ingestion, Espresso, *ts*) with (User, Ingestion, Tequila, *ts*), so that the false question is "Have I ever been to the Eiffel Tower to drink Tequila?"

The process mentioned above generates answerable questions. Similar to the idea of generating false questions, unanswerable questions are generated by replacing one of the events or the time of the events in answerable questions. For example, replacing event B of the second row in Table 1, the unanswerable question is "Where did I go after I went to the Champ de Mar to drink Tequila?" In order to avoid the problem of generating questions that users would never ask, we generate the unanswerable questions based on a principle that a question should contain at least one entity selected from the KB. That is, we assume that the question asked by the user in real practice contains at least one event that has been experienced before. In this way, the unanswerable questions are not generated from simple questions. The annotators are asked to check if the new combination is definitely different from the original one and unrelated to what the user has experienced.

In summary, there are two types of answerable questions: (1) the answers can be retrieved from the KB, and (2) the questions can be answered with "True" or "False" based on the KB. The rest of the questions are unanswerable. As a result, the dataset contains 41,960 distinct questions, where the numbers of answerable and unanswerable questions are 38,201 and 3,759, respectively. Table 2 shows the distribution of the seven question types with different combinations.

Combination	*Ea*	*EaEb*		*EaEbEc*	
Answerable	Yes	Yes	No	Yes	No
when	6,238	6,512	916	2,095	279
where	1,218	1,808	300	1,617	330
what	1,636	9,758	1,139	3,376	614
who	274	627	74	310	62
how many	20	114	8	55	10
duration	5	74	9	110	18
T/F	572	1,308	0	474	0

Table 2: The distribution of questions in our dataset.

Personal Knowledge Base Question Answering with Correction Mechanism

Figure 2 shows an overview of our PKBQAC system. The PKBQAC system is composed of two stages. In the first stage, the question answering model is aimed at generating candidate query graphs to extract the answer from the KB for the input question. If no answer can be found, it is recognized as an unanswerable question. After correcting query graphs, the question generation model in the second stage generates fluent questions with question editing. We will elaborate each stage in the following sections in detail.

[2] http://nlg.csie.ntu.edu.tw/nlpresource/PKBQAC_dataset/

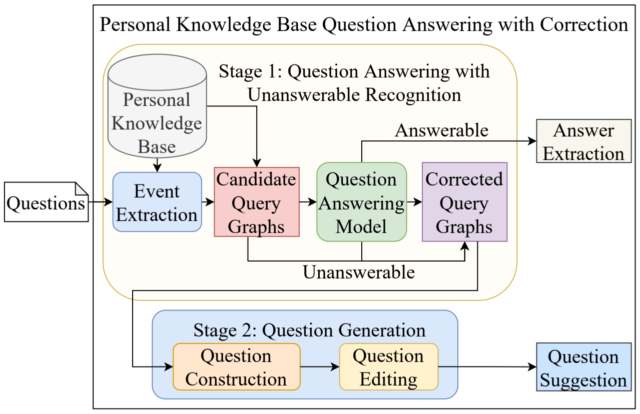

Figure 2: Overview of Our PKBQAC System.

First Stage: Question Answering with Unanswerable Recognition

We propose a question answering model based on semantic parsing, which aims to learn query graphs for representing semantic structures of a question. The candidate query graph is formed as a tree graph, where the root of the tree is the answer node representing the real answer entity. The other nodes in the tree are the entities extracted from the question, which constrain the answer node by relations in the KB. We compute the semantic similarity between the question and each candidate query graph. The query graph with the highest score will be reported as the answer.

Event Extraction For generating the candidate query graphs, we propose a text tagger based on the BERT model with condition random field layer to extract predicates and entity mentions in the question and transform them to the events in the form of (subject, predicate, object, time). The tasks of entity mention and predicate extraction are viewed as the problem of sequence labeling with the BIO scheme.

Since our dataset contains seven types of questions, another classifier, also based on BERT, is constructed to classify the type of a question. This task is regarded as a multi-class classification. Given a question q, our goal is to develop a model to predict the category of q, formulated as $\arg\max_{y \in Q} P(y|q)$, where Q is the set of seven question types in our dataset. We use the softmax layer to compute the probabilities of the seven question types.

Query Graph Generation Considering an entity having multiple surface forms, and the gap between natural language predicate and KB relation, we propose another BERT-based model to align the extracted predicates and mentions to the relations and entities in the personal KB, respectively. Formally, we align those extracted entity mentions m to the entities e in KB by using the BERT model, where m and e are represented by 128 tokens. They are input to the BERT model as a sequence pair with start and separator tokens: ([CLS] m [SEP] e [SEP]). The process of aligning those extracted predicates to the relations in KB is the same as the method described above. For the query graph generation, we view the entities as nodes, and the relation is the edge connected with two nodes. We perform breadth-first search (BFS) to find the entities connected with e for generating n candidate query graphs $\{g_1, g_2, ..., g_n\}$. For example, a candidate query graph of the second question in Table 1 is shown in Figure 3. The solid arrow and solid circle denote the events identified from the question. The dotted arrow denotes the possible path connected with the identified events.

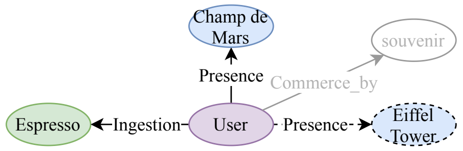

Figure 3: Example of a Candidate Query Graph.

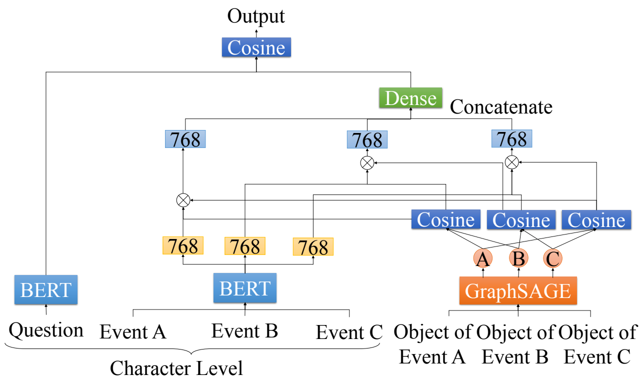

Figure 4: Structure of Our Question Answering Model.

Question Answering Model Figure 4 shows the neural network architecture of our question answering model. At the word-level, we encode a question and a query graph by two BERT models. Note that we assume a query graph consists of at most three events from the personal KB. When a query graph consists of only two events, the third input event is represented as a zero vector.

Graph neural networks have proven to be effective in question answering tasks (Dhingra et al. 2018; Qiu et al. 2019). In addition to semantic representations, the node embeddings generated by GraphSAGE (Hamilton, Ying, and Leskovec 2017) are incorporated into our question answering model to enrich features. GraphSAGE is an inductive framework that leverages features from a node's local neighborhood to generate embeddings. We utilize GraphSAGE to obtain the embedding of each node in the KB. We compute the cosine similarity between object nodes of events A, B and C. The similarity scores are used as the weight of each event:

$$w_A = \cos(v_A, v_B) + \cos(v_A, v_C) \tag{1}$$

$$w_B = \cos(v_A, v_B) + \cos(v_B, v_C) \tag{2}$$

$$w_C = \cos(v_A, v_C) + \cos(v_B, v_C) \tag{3}$$

$$X_g = [w_A \times h_A; w_B \times h_B; w_C \times h_C] \tag{4}$$

$$z_g = \tanh(W_g \cdot X_g + b_g) \tag{5}$$

$$score_{q,g} = \cos(h_q, z_g) \tag{6}$$

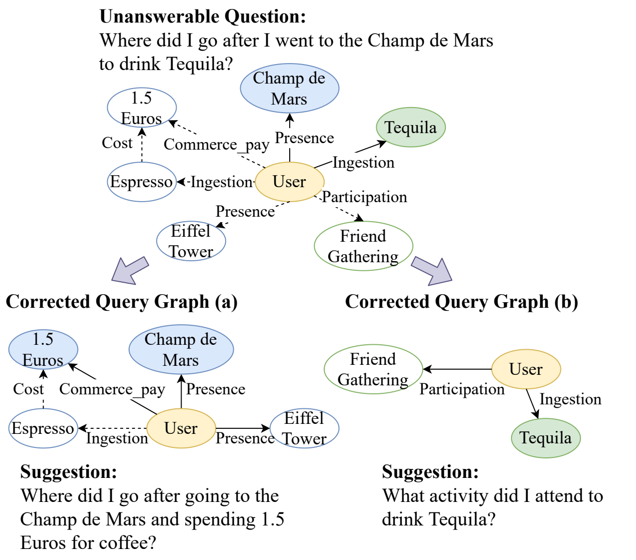
Unanswerable Question:
Where did I go after I went to the Champ de Mars to drink Tequila?
1.5 Euros
Champ de Mars
Tequila
Cost
Commerce_pay
Presence
Ingestion
Espresso
Ingestion
User
Participation
Presence
Eiffel Tower
Friend Gathering
Corrected Query Graph (a)
Corrected Query Graph (b)
Suggestion:
Where did I go after going to the Champ de Mars and spending 1.5 Euros for coffee?
Suggestion:
What activity did I attend to drink Tequila?

Figure 5: Example of Corrected Query Graphs from Unanswerable Question.

where v_α is node embedding of the object in the event α generated from GraphSAGE, α denotes events of A, B or C, and h_α is the final hidden state output from the BERT of input event α. We multiply the hidden state of each event h_α by its own weight w_α as the feature representation. We then obtain the final representation z_g of the candidate query graph g by concatenating the feature representation of each event and connecting with a dense layer. If the events are unrelated, the weights of events will be low. That makes the event vectors no contribution to the dense layer. The notation $score_{q,g}$ denotes the similarity score between the final hidden state of question h_q and z_g. We refer this model as Graph Weighted Semantic Matching Network (GWSMNet).

If the question is predicted as the type of "who", "what", "where", or "when", the highest similarity score of the candidate query graph g is extracted as the answer. If the question is predicted as the type of "how many" or "duration", we compute the frequency or duration of the events given the candidate query graph with the highest similarity score. For the question predicted as the type of "True or False", the system returns "True" if the highest similarity score is higher than 0.5. Otherwise, the system returns "False". If the type of question is not predicted as "True or False" and the highest similarity score is lower than 0.5, the question is regraded as unanswerable.

For those questions that cannot be answered, our goal is to generate questions related to what the user may want to ask. Figure 5 shows an example of two corrected query graphs g' of an unanswerable question. The blue and green circles represent the two different events of the user, which are irrelevant to each other. The circles filled with colors denote the entities mentioned in the question. Formally, the personal knowledge graph of the user is transformed to an adjacency matrix A. We find out all possible k-length paths from the given entity node to the connected node as corrected query graphs by computing A^k. Each corrected query graph g' is input to the question generation model for converting to a natural language question.

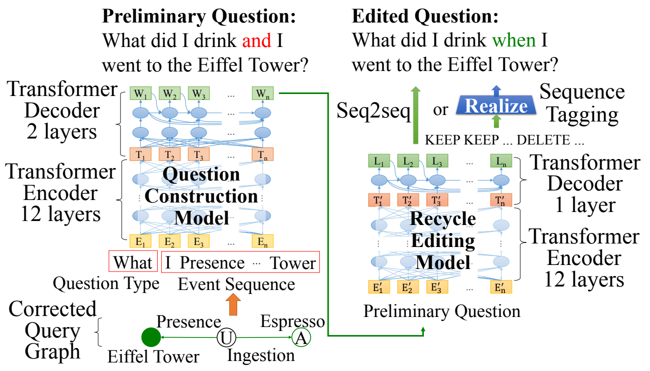

Figure 6: Structure of Question Generation Model.

Second Stage: Question Generation

We propose a RL based question generation model with question editing to generate feasible questions. It consists of a question construction model and a question editing model. The network structure is shown in Figure 6.

RL Based Question Generation Model with Question Editing The question construction model takes a corrected query graph g' and the question type as input and constructs a natural language question $\hat{q}$. The question type is taken as input to make the types of corrected questions consistent with those of unanswerable questions. First, we convert events in the corrected query graph to a word sequence that is referred as the event sequence hereafter. Taking the corrected query graph (b) in Figure 5 as an example, the event sequence is "I Presence Eiffel Tower I Ingestion Espresso." Given the event sequences and question types, the question construction model generates preliminary questions. Since there may be grammatical errors in the questions, we propose a question editing model based on the sequence-to-sequence (seq2seq) architecture for correcting the preliminary questions. The question generation model is called **DualGenNet** to specify that dual generators are used.

The training process of the RL based question generation model with question editing is detailed in Algorithm 1. At each epoch, we first input n examples of event sequences and question types to the question construction model for generating preliminary questions q'. We utilize log-likelihood as the objective function C_{loss} of the question construction model to update parameters.

The question editing model takes q' as input to generate an edited question $\hat{q}$ as the final outcome. In order to reward the question editing model for refining questions to make it better than the preliminary questions, we first evaluate fluency using BLEU and evaluate answerability using the score proposed by Nema and Khapra (2018). The reward function is defined in Equation 7.

$$R(.) = \text{Answerability} + \text{BLEU4} \tag{7}$$

Algorithm 1: Training process of question generation model.

```
Input: Question construction model C, Question
  editing model E, Event sequences with their
  corresponding question types S
Output: Edited question $\hat{q}$
for number of epochs do
    for number of batches do
        Sample n examples from S;
        Use C to generate $q'$;
        Update C parameters by $C_{loss}$;
        Use E to edit $q'$ and generate $\hat{q}$;
        Use the outputs of C and E to calculate
          reward in Equation 7;
        Update E parameters by $E_{loss}$;
    end
end
```

The reward function is computed by comparing the questions before and after the refinement. In this way, the question editing model will learn how to edit the preliminary question generated by the question construction model in the training process, and make the final generated question more fluent.

Then, we use the "REINFORCE with a baseline" algorithm (Williams 1992) to reward the question editing model. The training loss E_{loss} combines both log-likelihood and reward score, defined as follows.

$$\begin{aligned} E_{loss} &= \lambda(R(\hat{q}) - R(q')) \\ &\quad + (1-\lambda)\sum_{t=1}^{T} \log p(w_t|w_{t-1}, ..., w_1, q') \end{aligned} \tag{8}$$

where $R(\hat{q})$ and $R(q')$ are the rewards of edited questions $\hat{q}$ and preliminary questions q' comparing with the reference questions written by annotators, respectively. $\lambda \in \{0, 1\}$ is a trade-off between log-likelihood and reward. After tuning λ by using validation data, we set λ to 0.6. To sum up, the log-likelihood score is used to measure whether the generated question is similar to the ground-truth. It is a basic function for training the sequence generation model. The reward score is used to measure whether the question editing model can refine the errors in the preliminary questions. It encourages the question editing model to generate more fluent questions.

For determining the hyperparameters of our question construction model and question editing model, we have experimented with the encoder and the decoder layers ranged from 1 to 12. Finally, the question construction model combines a BERT encoder with 12 self-attention layers, and a 2-layer Transformer decoder on the top of the BERT encoder. In the question editing model, the encoder is a 12-layer BERT-base model, and the decoder is a single-layer Transformer decoder. We use Adam optimizer with a learning rate of 0.002 and train our models for 20 epochs.

Sequence Tagging as Question Editing Inspired by LASERTAGGER (Malmi et al. 2019), a model for text editing by tagging three operations of keeping, deleting, and adding a word, we propose another question editing model based on the sequence tagging architecture. We refer this question generation model as **GenTagNet**, which uses a sequence tagging model after a generator. In GenTagNet, the editing tags generated by a tag labeler is used as ground truth for training. After obtaining a predicted tag sequence from the question editing model, we realize it to a question.

Our tag labeler generates a tag sequence by assigning a tag for each token in the preliminary question that is the same as LASERTAGGER: KEEP, DELETE, and ADD word. KEEP and DELETE indicate whether to keep the token in the output question. ADD word indicates which word x should be inserted to the preliminary question. x belongs to a Vocabulary *W* which is constructed by sorting the words in our dataset based on their frequencies and picking the top-N words that can cover 80% of tokens. A word not in *W* will not be put into the tag sequence. Although this may lead to incomplete questions, most of the questions generated by the question editing model are more reasonable than the preliminary questions. The benefit of tagging edit operations is that we can train the model with a much smaller vocabulary when only a small amount of human-annotated dataset is available.

Pre-training Question Editing Model In addition to train the question editing model only on our dataset, we pre-train the model on the ClueWeb09 corpus (Callan et al. 2009). We first collect questions from the ClueWeb09 corpus. Then, we produce ungrammatical questions that would be generated by the question construction model. Specifically, we manually produce ungrammatical questions based on our observations on the generated questions by the question construction model. The main error patterns in the ungrammatical questions are repeated phrases and improper wording. Therefore, we produce ungrammatical questions by paraphrasing questions from the external corpus according to the observed error patterns. Finally, we construct one million (ungrammatical question, correct question) pairs for pre-training the question editing model, where the correct question is the ground truth of the ungrammatical question. In other words, the ground-truth grammatical questions are the original questions in the ClueWeb09 corpus. The correct questions in sequence tagging model and seq2seq model are the tag sequences from the tag labeler and the original questions in the corpus, respectively.

Experiments

In this section, we evaluate the performances of the question answering model and the question generation model in our PKBQAC system, respectively. The best performance is in bold.

Evaluation of the Question Answering Model with Unanswerable Recognition

This section presents the performance of our question answering model. The sizes of the training, validation, and test sets are 18,137, 9,103, and 14,720 questions, respectively. The test set consists of 13,415 answerable questions

	Answerable			Unanswerable
Model	hit@1	hit@3	MRR	Accuracy
Luo et al. (2018)	74.35	74.56	0.7447	41.08%
BERT	75.93	76.14	0.7606	41.95%
GMNet (mean)	71.19	71.31	0.7127	50.82%
GMNet (LSTM)	71.34	71.48	0.7144	60.14%
GMNet (meanpool)	71.06	71.22	0.7117	50.02%
GMNet (maxpool)	71.30	71.45	0.7141	53.98%
GWSMNet (mean)	78.69	81.9	0.8033	69.82%
GWSMNet (LSTM)	**80.27**	**83.91**	**0.8212**	**69.89%**
GWSMNet (meanpool)	80.07	83.71	0.8192	68.79%
GWSMNet (maxpool)	78.51	82.00	0.8027	69.67%

Table 3: Performance on question answering with unanswerable recognition.

and 1,305 unanswerable questions. The performances are shown in Table 3. We adopt hit@1, hit@3, and mean reciprocal rank (MRR) for answerable questions, and accuracy for unanswerable questions.

Three models based on features at different levels are compared. We reproduce the model proposed by Luo et al. (2018) on our dataset as baseline model. The model proposed by Luo et al. (2018) and BERT are semantic matching network that only computes the similarity between the question and the candidate query graph at the semantic level. Graph Matching Network (GMNet) extracts the answer by computing the similarity between semantic features of the question and node embeddings in the corrected query graph. Graph Weighted Semantic Matching Network (GWSMNet) combines features at both the semantic and the graph levels to extract the answer. We also compare the performances of using different aggregator funtions in GraphSAGE which include mean operator, LSTM architecture, and pooling approach. We observe that GMNet with LSTM aggregator is better than with other aggregators. The reason may be that the LSTM aggregator has the advantage of the larger expressive capability.

In Table 3, GWSMNet with LSTM aggregator achieves an MRR score of 0.8212 on answerable questions and an accuracy of 69.89% on unanswerable questions, and significantly outperforms all baseline models with $p < 0.001$ using the McNemar's test. We find that incorporating semantic and graph level information is able to enrich feature representations and improve performances of both answerable and unanswerable questions.

Evaluation of the Question Generation Model

Our dataset contains 32,425 query graphs for the answerable questions, and 3,514 corrected query graphs for the unanswerable questions. The corrected query graphs are collected by setting the power of the adjacency matrix k to 2 for extracting 2-length paths from the entity to the connected entities. For evaluating the question editing model, we use 3,514 corrected query graphs as test data. In answerable questions, 90% of query graphs are used for training,

Model	BLEU4	ROUGE-L	QBLEU4
RefNet (Nema et al. 2019)	37.09	54.74	50.93
Transformer	40.01	60.10	54.49
DualGenNet w/o pre-train	42.38	61.68	56.74
GenTagNet w/o pre-train	43.01	63.10	57.49
DualGenNet w/ pre-train	43.39	64.71	57.83
GenTagNet w/ pre-train	**45.18**	**66.75**	**59.64**

Table 4: Performance of question generation.

and 10% for validation. Table 4 shows the performances of the question generation models.

In the experiments, we employ the beam search with a beam size of 3. The models are evaluated based on BLEU4, ROUGE-L, and Q-BLEU4 (Nema and Khapra 2018). We use Transformer as a seq2seq baseline model. Both of the encoder and the decoder are the BERT-based model. The latest question generation model RefNet (Nema et al. 2019) is also reproduced on our dataset as a baseline model. For applying RefNet to our dataset, we input the answer entity to the answer encoder, and input the word sequence of the remaining corrected query graph to the passage encoder.

The impact of pre-training the question editing model is also confirmed. In Table 4, the performances of the RL based models are better than all baseline models. Comparing DualGenNet and GenTagNet, we find that utilizing sequence tagging as the question editing model is better than using seq2seq model. Moreover, pre-training the question editing model improves the performance.

We also evaluate the performance of the pipeline workflow by combining the best models in the two stages, GWSMNet (LSTM) for question answering and GenTagNet for question generation. The performance of the question generation model is measured by $\frac{\sum_{i=1}^{M} f(gq_i, gt_i)}{M}$, where M is the number of unanswerable questions correctly predicted by GWSMNet (LSTM), gq_i is the i-th question generated by GenTagNet, gt_i is the ground-truth question, and f is the metric (e.g., BLEU4). With the impact of error propagation, the BLEU4, ROUGE-L, and QBLEU4 of GenTagNet are 44.72, 65.31, and 59.11, respectively.

Discussion

In this section, we first analyze the performances of the question answering model on different question types of answerable and unanswerable questions. Then, we analyze the performances of the question generation model trained in the pipeline workflow and in the end-to-end fashion. Finally, We analyze the impacts of the question generation model with different reward functions.

Evaluation of Question Answering Models on Different Question Types

As shown in Table 2, the distribution of question types is imbalanced. We further analyze the performances of the question answering model in different question types. The

Type	Frequency	Luo et al. (2018)	BERT	GMNet	GWSMNet
when	14,845	76.03%	76.48%	74.71%	80.57%
where	4,643	68.90%	69.85%	53.87%	80.34%
what	14,770	74.40%	77.83%	73.75%	80.90%
who	1,211	76.14%	74.40%	73.82%	77.29%
how many	189	41.80%	42.33%	48.68%	63.49%
duration	189	38.62%	41.27%	46.03%	52.91%
T/F	2,354	78.76%	78.76%	72.01%	79.44%

Table 5: Performance of Question Answering on Different Question Types of Answerable Questions.

Type	Frequency	Luo et al. (2018)	BERT	GMNet	GWSMNet
when	1,195	41.76%	42.18%	65.19%	74.90%
where	630	50.16%	49.84%	65.40%	79.05%
what	1,753	36.91%	37.99%	54.54%	63.15%
who	136	51.47%	58.09%	67.65%	75.74%
how many	18	27.78%	22.22%	50.00%	55.56%
duration	27	25.93%	37.04%	48.15%	51.85%

Table 6: Performance of Question Answering on Different Question Types of Unanswerable Questions.

performances for answerable and unanswerable questions are shown in Table 5 and Table 6, respectively. We compare the performances of the models proposed by Luo et al. (2018), BERT, GMNet, and GWSMNet, where the aggregators of GMNet and GWSMNet are LSTM. Experimental results show that the imbalance of different question types does affect the performance of the models. However, we find that all models achieve promising performances on the question types "T/F" compared with the two larger question types "when" and "what". Besides, the models incorporated with graph level information, i.e., GMNet and GWSMNet, achieve better performances on the question types of "how many" and "duration", which require inference between multiple events in the KB.

Evaluation of Pipeline Models and End-to-End Model in Question Generation

The RL based models, such as DualGenNet and GenTagNet, are end-to-end trainable models. We implement transformer and LASERTAGGER in pipeline workflow as baseline models. That is, we regard question construction and question editing as two sub-tasks in the question generation model. Table 7 shows the performances of the models in the pipeline workflow and in the end-to-end fashion. The model followed by the * symbol denotes the model trained in the pipeline workflow. Experimental results show that using the REINFORCE with a baseline algorithm to reward the question editing model based on the output from the question construction model can benefit the final results.

The human evaluation is also conducted to validate the measurement. We randomly sampled 100 corrected query graphs from our dataset. Questions generated from models in the two sub-tasks are shown to three annotators, and they are asked to select which one is better considering completeness, fluency, and answerability. We report the evaluation results that annotators prefer the generated questions by each model. In the question construction and the question editing, annotators prefer GenTagNet, especially in the question editing.

Subtask	Model	BLEU4	ROUGE-L	QBLEU4	Human
Question Construction	Transformer*	40.01	60.10	54.49	31.5%
	DualGenNet	40.98	61.85	55.53	33.4%
	GenTagNet	41.94	63.14	56.53	35.1%
Question Editing	Transformer*	41.38	62.23	55.85	13.4%
	LaserTagger*	42.26	63.51	56.83	15.7%
	DualGenNet	43.39	64.71	57.83	27.4%
	GenTagNet	**45.18**	**66.75**	**59.64**	**43.5%**

Table 7: Performance of models in the pipeline workflow and end-to-end fashion.

Reward	BLEU4	ROUGE-L	QBLEU4	Human
Answerability+BLEU4	**45.18**	**66.75**	**59.64**	**55.1%**
Answerability	44.96	66.39	59.41	22.7%
BLEU4	44.02	65.42	58.59	15.3%
N/A	43.98	65.33	58.60	6.9%

Table 8: Performances of Different Reward in GenTagNet.

Performances of Reward Function

We use the reward function in the question generation model, which combines the answerability score and BLEU. The answerability score measures whether the question contains question type, entities, and relations. In contrast, the BLEU score measures the overlaps of n-grams in candidate and reference sentences. In this section, we perform an ablation study to analyze the impact of reward function. In Table 8, we compare our question generation model GenTagNet under different reward functions. N/A denotes no reward function is used in the model.

Table 8 shows that the performances degrade when both reward functions for answerability and fluency are not considered at the same time. We randomly sample 100 corrected query graphs and show the questions generated by GenTagNet with four reward functions to three annotators. The annotators are asked to select which one is better. As a result, annotators prefer the GenTagNet model using both answerability and fluency as the reward function.The evaluation result shows the answerability score can better evaluate whether the generated questions are fluent and reasonable.

Conclusion

In this paper, we focus on developing a complete system for people querying their past experiences, which can be viewed as an application of PKBQA. Since people often muddle up events and ask questions that cannot be answered over personal KB, we address the topic of correcting unanswerable question based on personal KB. We propose a system consisting of a question answering model and a question generation model. The question answering model extracts answers based on personal KB by generating candidate query graphs. For the unanswerable questions, the corrected query graphs related to the user's intent will be suggested. The question generation model takes the corrected query graphs as input to generate answerable questions in natural language. Experimental results show the promising results.

In this work, our system corrects unanswerable questions for helping users query the facts they want to know in a single-turn only. Correcting unanswerable question with user interactions is a challenging issue to be resolved in the future.

Acknowledgments

This research was partially supported by Ministry of Science and Technology, Taiwan, under grants MOST-109-2634-F-002-040-, MOST-109-2634-F-002-034-, MOST-110-2634-F-002-028-, and MOST-109-2218-E-009-014-.

References

Auer, S.; Bizer, C.; Kobilarov, G.; Lehmann, J.; Cyganiak, R.; and Ives, Z. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*, 722–735. Springer.

Bao, J.; Duan, N.; Yan, Z.; Zhou, M.; and Zhao, T. 2016. Constraint-based question answering with knowledge graph. In *COLING*, 2503–2514.

Bollacker, K.; Evans, C.; Paritosh, P.; Sturge, T.; and Taylor, J. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, 1247–1250.

Bordes, A.; Usunier, N.; Chopra, S.; and Weston, J. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075* .

Buck, C.; Bulian, J.; Ciaramita, M.; Gajewski, W.; Gesmundo, A.; Houlsby, N.; and Wang, W. 2018. Ask the Right Questions: Active Question Reformulation with Reinforcement Learning. In *International Conference on Learning Representations*.

Bush, V. 1945. As we may think. *The atlantic monthly* 176(1): 101–108.

Callan, J.; Hoy, M.; Yoo, C.; and Zhao, L. 2009. Clueweb09 data set.

Cathal Gurrin, Hideo Joho, F. H. L. Z. V.-T. N. T.-K. L. R. A. D. T. D. N.; and Healy, G. 2019. Overview of NTCIR-14 Lifelog-3 task. NTCIR.

Chen, Y.; Wu, L.; and Zaki, M. J. 2019. Reinforcement Learning Based Graph-to-Sequence Model for Natural Question Generation. In *International Conference on Learning Representations*.

Christmann, P.; Saha Roy, R.; Abujabal, A.; Singh, J.; and Weikum, G. 2019. Look before you hop: Conversational question answering over knowledge graphs using judicious context expansion. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 729–738.

Dang-Nguyen, D.-T.; Piras, L.; Riegler, M.; Tran, M.-T.; Zhou, L.; Lux, M.; Le, T.-K.; Ninh, V.-T.; and Gurrin, C. 2019. Overview of ImageCLEFlifelog 2019: solve my life puzzle and lifelog moment retrieval. In *CLEF2019 Working Notes. CEUR Workshop Proceedings*, volume 2380, 09–12.

Dhingra, B.; Jin, Q.; Yang, Z.; Cohen, W.; and Salakhutdinov, R. 2018. Neural Models for Reasoning over Multiple Mentions Using Coreference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, 42–48.

Dong, L.; Mallinson, J.; Reddy, S.; and Lapata, M. 2017. Learning to Paraphrase for Question Answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 875–886.

Dubey, M.; Banerjee, D.; Abdelkawi, A.; and Lehmann, J. 2019. Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia. In *International Semantic Web Conference*, 69–78. Springer.

Elsahar, H.; Gravier, C.; and Laforest, F. 2018. Zero-Shot Question Generation from Knowledge Graphs for Unseen Predicates and Entity Types. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 218–228.

Gurrin, C.; Joho, H.; Hopfgartner, F.; Zhou, L.; and Albatal, R. 2016. Ntcir lifelog: The first test collection for lifelog research. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, 705–708.

Gurrin, C.; Joho, H.; Hopfgartner, F.; Zhou, L.; Gupta, R.; Albatal, R.; Nguyen, D.; and Tien, D. 2017. Overview of ntcir-13 lifelog-2 task. NTCIR.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, 1024–1034.

Hu, M.; Wei, F.; Peng, Y.; Huang, Z.; Yang, N.; and Li, D. 2019. Read+ verify: Machine reading comprehension with unanswerable questions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 6529–6537.

Iyyer, M.; Yih, W.-t.; and Chang, M.-W. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1821–1831.

Jiang, L.; Liang, J.; Cao, L.; Kalantidis, Y.; Farfade, S.; and Hauptmann, A. 2017. Memexqa: Visual memex question answering. *arXiv preprint arXiv:1708.01336* .

Luo, K.; Lin, F.; Luo, X.; and Zhu, K. 2018. Knowledge base question answering via encoding of complex query graphs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2185–2194.

Malmi, E.; Krause, S.; Rothe, S.; Mirylenka, D.; and Severyn, A. 2019. Encode, Tag, Realize: High-Precision Text Editing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th*

International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 5057–5068.

Nema, P.; and Khapra, M. M. 2018. Towards a Better Metric for Evaluating Question Generation Systems. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 3950–3959.

Nema, P.; Mohankumar, A. K.; Khapra, M. M.; Srinivasan, B. V.; and Ravindran, B. 2019. Let's Ask Again: Refine Network for Automatic Question Generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 3305–3314.

Qiu, L.; Xiao, Y.; Qu, Y.; Zhou, H.; Li, L.; Zhang, W.; and Yu, Y. 2019. Dynamically fused graph network for multi-hop reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 6140–6150.

Rajpurkar, P.; Jia, R.; and Liang, P. 2018. Know What You Don't Know: Unanswerable Questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 784–789.

Sorokin, D.; and Gurevych, I. 2018. Modeling Semantics with Gated Graph Neural Networks for Knowledge Base Question Answering. In *Proceedings of the 27th International Conference on Computational Linguistics*, 3306–3317.

Sun, F.; Li, L.; Qiu, X.; and Liu, Y. 2018. U-net: Machine reading comprehension with unanswerable questions. *arXiv preprint arXiv:1810.06638* .

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4): 229–256.

Xiong, W.; Yu, M.; Chang, S.; Guo, X.; and Wang, W. Y. 2019. Improving Question Answering over Incomplete KBs with Knowledge-Aware Reader. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 4258–4264.

Xu, J.; Wang, Y.; Tang, D.; Duan, N.; Yang, P.; Zeng, Q.; Zhou, M.; and Xu, S. 2019. Asking Clarification Questions in Knowledge-Based Question Answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 1618–1629.

Yahya, M.; Barbosa, D.; Berberich, K.; Wang, Q.; and Weikum, G. 2016. Relationship queries on extended knowledge graphs. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, 605–614.

Yen, A.-Z.; Huang, H.-H.; and Chen, H.-H. 2019a. Multimodal joint learning for personal knowledge base construction from Twitter-based lifelogs. *Information Processing & Management* doi:10.1016/j.ipm.2019.102148.

Yen, A.-Z.; Huang, H.-H.; and Chen, H.-H. 2019b. Personal Knowledge Base Construction from Text-based Lifelogs. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 185–194.

Zhao, W.; Wang, L.; Shen, K.; Jia, R.; and Liu, J. 2019. Improving Grammatical Error Correction via Pre-Training a Copy-Augmented Architecture with Unlabeled Data. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 156–165.