Automatically Generating Questions from Queries for Community-based Question Answering

Shiqi Zhao[1,2], **Haifeng Wang**[1], **Chao Li**[2], **Ting Liu**[2], **Yi Guan**[2]
[1]Baidu, Beijing, China
{zhaoshiqi,wanghaifeng}@baidu.com
[2]Harbin Institute of Technology, Harbin, China
beyondlee2008@yahoo.cn, tliu@ir.hit.edu.cn, guanyi@hit.edu.cn

Abstract

This paper proposes a method that automatically generates questions from queries for community-based question answering (cQA) services. Our query-to-question generation model is built upon templates induced from search engine query logs. In detail, we first extract pairs of queries and user-clicked questions from query logs, with which we induce question generation templates. Then, when a new query is submitted, we select proper templates for the query and generate questions through template instantiation. We evaluated the method with a set of short queries randomly selected from query logs, and the generated questions were judged by human annotators. Experimental results show that, the precision of 1-best and 5-best generated questions is 67% and 61%, respectively, which outperforms a baseline method that directly retrieves questions for queries in a cQA site search engine. In addition, the results also suggest that the proposed method can improve the search of cQA archives.

1 Introduction

In recent years, community-based question answering (cQA) services become popular, such as Yahoo! Answers (answers.yahoo.com) in English and Zhidao (zhidao.baidu.com) in Chinese. In cQA, people can have their questions answered by other people rather than by automatic QA systems, which usually better guarantee the answer quality. Till Oct. 2010, Zhidao has accumulated over 100 million answered questions, which form an extremely large and valuable knowledge base.

Lin (2008) first proposed the idea of automatically generating questions from queries. The underlying assumption is that when a user issues a query to a search engine, he could have a question in mind but it is more convenient and efficient for him to realize the question as a query. This technique could have a great impact on cQA services.

First, it can improve the search of cQA archives. As we know, most of cQA resources can be searched with general search engines like Google and Baidu (www.baidu.com). Many of them also have their own site search engines. However, since user queries are mostly short and incomplete, it is quite often that many less relevant questions are retrieved when searching cQA archives. By generating questions from short queries, we can expand the queries and estimate questions that are more likely to be interested in, which could help to retrieve more related questions from cQA archives.

Second, this technique can be useful in enlarging cQA resources. For example, in some search engines like Baidu, if a query is found to be frequently searched, the query will be automatically submitted to a cQA site and expected to be answered by some users. However, a problem is that the frequent queries are usually short and incomprehensible, which makes it hard for users to understand and answer the queries. If we can generate questions from queries, it will become more natural for users to answer the questions, thus contribute new data to the cQA resources[1].

Third, the technique can also be used in query analysis. Through reformulating queries to well-formed questions, it will get easier to analyze the relationship of the query terms as well as the focus of the users' requirement.

In this paper, we follow the proposal of (Lin, 2008) and put forward a novel method for query-to-question generation. The method includes two stages, i.e., template acquisition and question generation. In the former stage, the method collects query-to-question pairs from search engine query

[1]The generated incorrect questions will be ignored by users, which will not evidently influence the performance.

Proceedings of the 5th International Joint Conference on Natural Language Processing, pages 929–937,
Chiang Mai, Thailand, November 8 – 13, 2011.

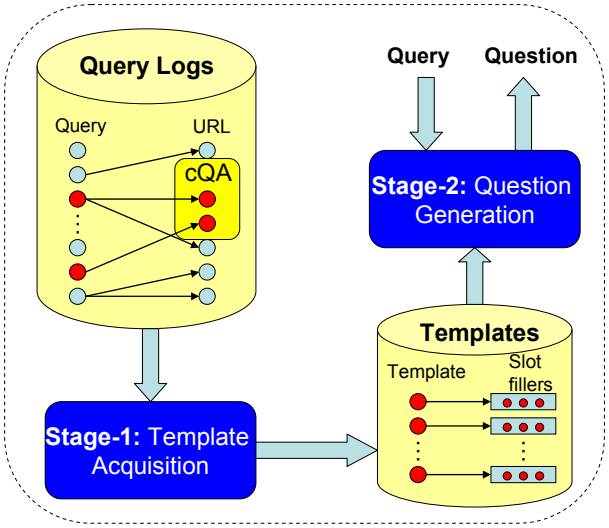

Figure 1: Overview of the method.

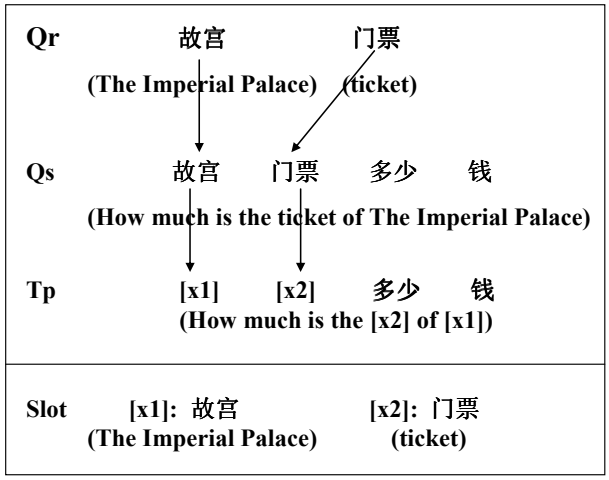

Figure 2: An example of Qr, Qs, and Tp.

logs and further extracts question generation templates. In the latter stage, it generates questions for input queries using the obtained templates. Figure 1 illustrates these two stages.

We conducted experiments on a set of 1000 short queries randomly selected from Baidu's query logs. The results show that our method is effective. Specifically, the method generates questions for 76.5% test queries. The precision of 1-best and 5-best questions is 67% and 61%, respectively, which evidently outperforms a baseline that directly retrieves questions for queries using a cQA site search engine. In addition, we designed a strategy to improve the search of cQA archives through query-to-question generation, and the result is quite promising. Although our experiments were carried out in Chinese, the method can be extended to other languages in which cQA archives exist.

2 Proposed Method

2.1 Template Acquisition

As mentioned above, the cQA archives are frequently searched and viewed through search engines. Therefore, we can find a large number of records in search engine query logs, in which users searched a query Qr and clicked on a question Qs from the cQA archives. Such $\langle Qr, Qs\rangle$ pairs can be collected and used for training question generation models. In our method, we mine $\langle Qr, Qs\rangle$ pairs from Baidu's query logs. In detail, suppose a user issued a query Qr in Baidu and clicked on a search result with the title T, then $\langle Qr, T\rangle$ will be extracted as a $\langle Qr, Qs\rangle$ pair if it meets the following constraints:

- Qr is not a question[2], and it contains at most three terms. Our intuition is that long queries might be clear enough which need not to be expanded into questions.
- T should be a question Qs and must be from a cQA web site (Zhidao in our experiments).
- Qr should be subsumed in Qs. This constraint limits that Qr can only be *extended* to questions, whose terms are not allowed to be transformed or deleted during question generation.

From the identified $\langle Qr, Qs\rangle$ pair, we can induce a template Tp by substituting query terms in the question Qs with slots $\{[x_i]|1 \leq i \leq n\}$, where n is the number of query terms of Qr. An example is shown in Figure 2. In what follows, we term Qs as an *instantiation* of Tp.

In our experiments, we obtained over 15 million $\langle Qr, Qs\rangle$ pairs from the query logs used, and accordingly extracted 547,325 templates. To eliminate templates that are rarely instantiated, we filtered those templates with less than 10 unique instantiations. 18,929 templates were left after filtering, each with 80 unique instantiations on average. Analysis result reveals that 80% of the eliminated templates are those long and complicated ones, which may seldom be used in practice.

[2]We developed a rule-based tool to identify whether a Chinese word sequence is a question.

Note that a query could instantiate more than one templates. For example, query "故宫 门票 (*The Imperial Palace / ticket*)" can instantiate "*[x1] [x2]* 多少 钱 (*How much is the [x2] of [x1]*)", "*[x1] [x2]* 价格 是 多少 (*What is the price of [x1]'s [x2]*)", and "*[x1] [x2]* 贵 吗 (*Is the [x2] of [x1] expensive*)", etc. We therefore need to compute the likelihood that a query instantiates each template, which can be used in template ranking and selecting during question generation. In our work, given query Qr and all templates it can instantiate $\{Tp_1, ..., Tp_n\}$, we compute the likelihood of Qr instantiating $Tp_i (1 \leq i \leq n)$ based on maximum likelihood estimation:

$$p(Tp_i|Qr) = \frac{c(Qr, Tp_i)}{c(Qr)} \quad (1)$$

where $c(Qr, Tp_i)$ is the frequency that Qr instantiates Tp_i in the query logs, $c(Qr)$ is the frequency that Qr occurs in the query logs. The acquired templates along with the queries that can instantiate each template are stored in a database D_{Tp}.

2.2 Question Generation

With the induced templates, we can generate questions for any input query qr[3]. Since most of the templates are unsuitable for qr, we need a strategy to select the templates and only retain the useful ones. Our observation is that similar queries usually have close search intent. They may tend to instantiate identical templates and generate similar questions. For example, we found that queries about the tickets of something are mostly interested in the price and instantiate template "*[x1] [x2]* 多少 钱 (*How much is the [x2] of [x1]*)". Thus when we get a new query qr about tickets, it is reasonable for qr to instantiate the same template and generate a question qs asking about the price.

Guided by this intuition, we generate questions for query qr using the templates of qr's similar queries. In practice, we first retrieve qr's similar queries from D_{Tp}, each of which must contain the same number of terms and share at least one identical term with qr. After that, we collect all templates that can be instantiated by the retrieved similar queries, which are then instantiated by qr to generate a list of questions. All the generated questions are ranked and the top-N are returned. The example in Figure 3 illustrates this process.

We take two factors into account when ranking the generated questions. The first is the likelihood that query qr instantiates template Tp, and the second is the fluency of the generated question qs. Hence we define:

$$\begin{aligned} \hat{qs} &= \arg\max_{qs} f(qs, Tp, qr) \quad (2) \\ &= \arg\max_{qs} \{\lambda f_{TP}(Tp, qr) + (1-\lambda) f_{LM}(qs)\} \end{aligned}$$

where $f(qs, Tp, qr)$ is the score function for question ranking, which is decomposed into two parts, i.e., $f_{TP}(Tp, qr)$ and $f_{LM}(qs)$. The former computes the likelihood that qr instantiates Tp, while the latter measures the fluency of qs generated by instantiating Tp with qr.

Definition of $f_{TP}(Tp, qr)$. Let $\{Qr_i|1 \leq i \leq I\}$ be the similar queries of qr that can instantiate template Tp, we define $f_{TP}(Tp, qr)$ as:

$$f_{Tp}(Tp, qr) = \log \sum_{i=1}^{I} p(Tp|Qr_i) p(Qr_i|qr) \quad (3)$$

where $p(Tp|Qr_i)$ is the probability that Qr_i instantiates Tp, which has been defined in Equ. (1). $p(Qr_i|qr)$ reflects the degree that qr resembles Qr_i, which is defined as the query similarity $sim(qr, Qr_i)$ and is computed as:

$$sim(qr, Qr_i) = \prod_{j=1}^{J} sim(t_{qr\_j}, t_{Qri\_j}) \quad (4)$$

where $sim(t_{qr\_j}, t_{Qri\_j})$ is the similarity between the j-th term of qr and Qr_i, and J is the number of terms in both qr and Qr_i[4]. According to Equ. (4), qr and Qr_i are deemed similar only when they are similar term by term. This guarantees that each term of qr can safely fill in the slot induced from the corresponding term of Qr_i. The similarity between two terms t_1 and t_2 is computed based on distributional hypothesis, which assumes that words occurring in similar contexts tend to have similar meanings (Harris, 1985). We therefore define $sim(t_1, t_2)$ as the similarity of the context words of the two terms:

$$sim(t_1, t_2) = \cos(V_{ctx}(t_1), V_{ctx}(t_2)) \quad (5)$$

[3]Here we use qr and qs to denote a new query and its generated question, so as to differentiate from Qr and Qs that represent query and question mined from query logs.

[4]Recall that each similar query of qr must have the same number of terms as qr.

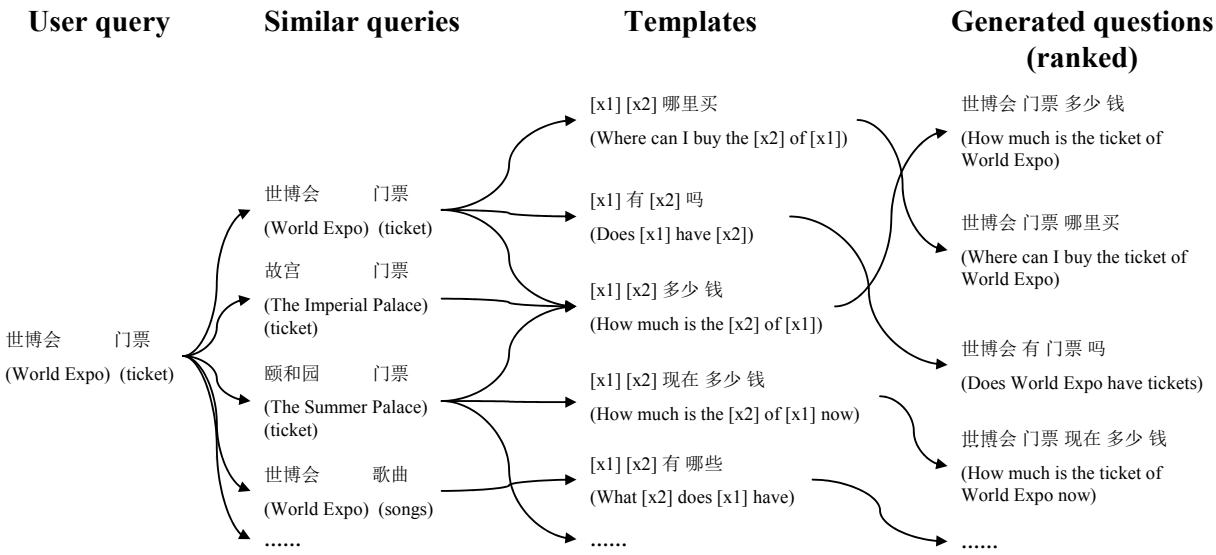

Figure 3: Generating questions from queries with templates induced from search engine query logs.

where $\cos(.,.)$ is the cosine similarity between two vectors. $V_{ctx}(t)$ is the vector of context words of t, which is constructed using Baidu query logs. Specifically, words occurring within the same queries as t are extracted as t's context words. The weight of each context word w is computed in a similar way as tf-idf:

$$W_t(w) = tf_t(w) \times \log \frac{N}{n(w)} \quad (6)$$

where $tf_t(w)$ is the frequency that w occurs in the contexts of t. $n(w)$ is the number of terms whose context words contain w. N is the total number of terms, i.e., the size of vocabulary.

Definition of $f_{LM}(qs)$. The other score function $f_{LM}(qs)$ is designed to measure the fluency of the generated questions, which is defined based on a tri-gram language model:

$$f_{LM}(qs) = \frac{1}{L} \log(p_{LM}(qs)) \quad (7)$$

in which L is the number of terms in qs, and $p_{LM}(qs)$ is the language model score of qs:

$$p_{LM}(qs) = \prod_{l=1}^{L} p(t_l | t_{l-2} t_{l-1}) \quad (8)$$

In our experiments, the language model was trained using over 15 million questions from the collected $\langle Q_r, Q_s \rangle$ pairs. We estimated the parameter λ in Equ. (2) using a development set with 247 random queries. In detail, we generated questions for each query and had all questions manually annotated (Section 3.1). We then examined λ ranging from 0 to 1 and evaluated $P@5$ precision (Section 3.2) under each setting. The setting that obtained the highest performance was selected, which was $\lambda = 0.3$.

3 Evaluation

Our experiments contain two parts. In the first part, we evaluated the precision of the generated questions based on human annotation. In the second part, we used the query-to-question generation algorithm to improve the search of cQA archives.

3.1 Experimental Setup

Comparison method. To our knowledge, there is no existing system that can automatically generate questions from queries. We therefore design a baseline method for comparison, which retrieves questions from cQA archives (termed as RcQA hereafter). Given a query qr, RcQA searches qr in Zhidao site search engine and retrieves questions containing qr[5]. The questions are ranked according to the orders assigned by the site search engine. We compare our query-to-question generation method (QtQG for short) and RcQA on precision. The reason why we employ the cQA site

[5]The retrieved titles from Zhidao are not necessarily questions. We ignored the non-question results when collecting questions from Zhidao.

search engine in the baseline instead of computing the similarity between queries and questions by ourselves is that we believe the cQA site search engine has adopted a state-of-the-art method when computing query-question similarity.

Experiment data. To construct a test set, we randomly sampled 1000 queries from Baidu query logs, each of which contains no more than 3 terms. The test queries cover a variety of domains, including *health*, *food*, *sport*, *music*, *movie*, *software*, *computer game*, etc. As described in Section 2, we used our method to generate questions for each test query, which were ranked according to Equ. (2). We kept up to top-5 questions of each query for evaluation. Meanwhile, we also retrieved up to top-5 questions for each query from Zhidao with RcQA.

Human annotation. Questions produced with both QtQG and RcQA were evaluated based on human annotation. We had two annotators, both of whom are native Chinese speakers. The questions produced with two methods were mixed before being presented to the annotators, so as to avoid bias during annotation. Two annotators evaluated the questions separately. For a query qr, a generated question qs is annotated as *correct* if it is fluent, comprehensible, and likely to be asked by people when they search qr. Otherwise, qs is annotated as *incorrect*. For instance, for the query "*世博会 门票* (*World Expo / ticket*)", question "*世博会 门票 多少钱* (*How much is the ticket of the World Expo*)" was annotated as correct whereas "*世博会 门票 哪里 可以 下载* (*Where can I download the ticket of the World Expo*)" was annotated as incorrect. In addition, a question was annotated as incorrect if it contains too much extra information that is impossible to be induced from the query[6]. For instance, question "*在 昆山 怎么 买 世博会 门票 (学生票)* (*How can I buy student tickets of Word Expo in Kunshan*)" was judged as incorrect for the above query, since "*学生票* (*student tickets*)" and "*在 昆山* (*in Kunshan*)" cannot be induced from the query.

After the first round of annotation, we calculated the kappa statistic between two annotators. The result shows that kappa $K = 0.78$, indicating a substantial agreement (K: 0.61-0.8) according to (Landis and Koch, 1977). Data with different annotations were then annotated by a third-party judge, so as to get the final annotations.

[6]Note that question descriptions written by question askers to further explain the questions are not extracted and evaluated together with questions, thus our evaluation metrics are not biased against questions with long and verbose descriptions.

	$P@1$	$P@2$	$P@3$	$P@4$	$P@5$
QtQG	0.67	0.66	0.64	0.62	0.61
RcQA	0.47	0.44	0.42	0.40	0.40

Table 1: $P@N$ results of QtQG and RcQA.

3.2 Evaluation of Precision

Experimental results show that, QtQG can generate at least one question for 765 queries, while RcQA can retrieve at least one question from Zhidao for 660 queries. It suggests that QtQG achieves a larger coverage than RcQA. After the questions were manually annotated, we computed the precision at top-N results ($P@N$):

$$P@N = \frac{|S_{CQs}(N) \cap S_{Qs}(N)|}{|S_{Qs}(N)|} \quad (9)$$

where $S_{Qs}(N)$ denotes the set of top-N questions and $S_{CQs}(N)$ denotes the set of correct ones. Table 1 summarizes the $P@N$ results ($1 \leq N \leq 5$) averaged on the test set for the two methods, from which we can see that QtQG significantly outperforms RcQA. Especially, we observed the results and found that QtQG can generate correct questions for some unpopular queries, about which we cannot even find questions from the web. For example, query "*昆明 KTV 沙发* (*Kunming / KTV / sofa*)" can retrieve no question from the web, but our method can generate question "*昆明 哪里 有 卖 KTV 沙发 的* (*Where to Buy KTV sofa in Kunming*)", which makes perfect sense and is quite likely to reflect users' requirement. Please note that some of the learned templates can reorder the query terms when generating questions. For example, for the query "*门票 颐和园* (*ticket / the Summer Palace*), the template "*[X2] 的 [X1] 多少钱*" can be matched and the correct question "*颐和园 的 门票 多少钱* (*How much is the ticket of the Summer Palace*)" can be generated.

We then conducted error analysis on the data set. Specifically, we sampled 100 incorrect questions produced with QtQG and RcQA, respectively. Analysis results reveal that 84% of incorrect questions generated with QtQG are those incomprehensible or not fluent questions, such as question "*德州 到 天气预报 怎么走* (*How can I get to weather forecast from Dezhou*)" for query "*德*

州 天气预报 (*Dezhou / weather forecast*)". The rest errors are questions that are well formed but less likely to be asked by people. On the other hand, for RcQA, 91% errors are questions containing too much extra information. For example, for query "*温州 歌曲* (*Wenzhou / songs*)", RcQA retrieves the question "*谁 能 提供 给 我 一些 关于 温州小吃 的 诗 歌， 歌曲 或者 趣味 知识* (*Who can give me some poems, songs, or interesting knowledge about Wenzhou snacks*)" . Obviously, it is too specific and not so good as the question generated with QtQG "*关 于 温州 的 歌曲 有 哪些* (*Are there any songs about Wenzhou*)".

Further more, we also analyzed queries for which QtQG or RcQA failed to generate questions. We sampled 100 such queries for the two methods and had them manually annotated. Results show that, 84% queries for which QtQG cannot generate questions are single-word queries. This is due to our limitation when searching similar queries for a new query qr (Section 2.2), that the similar queries for qr must share at least one identical word with qr. This constraint is designed to restrict search space when looking for similar queries. However it also limits that a single-word query can find similar queries and be rewritten as questions only when the same query has appeared in the D_{Tp} database (Section 2.1). This is a drawback of our method, which will be addressed in the future work. On the other hand, 82% queries for which RcQA did not retrieve questions are because the queries are unpopular. No question about them has been asked in Zhidao.

3.3 Improving cQA Search

As mentioned above, one application of query-to-question generation is to improve the search of cQA archives. We therefore carried out an experiment to examine its effectiveness. We design a strategy to integrate the QtQG module into the cQA site search engine. The basic idea is that, given a query qr, its generated question qs, and a cQA site search engine CQA, if qr's search result in CQA is unsatisfactory, but qs's result is good, then we can return qs's result for qr. The strategy is formally described in Table 2. The key problem here is how to estimate the quality of the search results. This is an interesting research topic but out of the scope of this paper. In our experiment, we adopted a simple criterion, namely, computing the similarity between the query and the title of the search result. The larger the similarity, the better the result. This criterion is naive, but our experiment result below shows that it is enough to verify the effectiveness of QtQG in cQA search.

Input:	qr: user query
	qs: generated question for qr
	CQA: cQA site search engine
Output:	$RST(qr)$: search result of qr in CQA
1.	Search qr in CQA, get result $R(qr)$
2.	Search qs in CQA, get result $R(qs)$
3.	Estimate the quality of $R(qr)$, get $E(R(qr))$
4.	Estimate the quality of $R(qs)$, get $E(R(qs))$
5.	**If** $E(R(qr)) = BAD$ and $E(R(qs)) = GOOD$
6.	Return $RST(qr) = R(qs)$
7.	**Else**
8.	Return $RST(qr) = R(qr)$

Table 2: Strategy for improving cQA search with automatically generated questions.

We experimented with the 765 test queries for which at least one question can be generated with our method. We only examined the 1-best question generated for each query. The cQA site search engine used is Zhidao. In addition, when evaluating and comparing the search results of the original query and generated question, we just considered the top-1 search result, which is not only for convenience, but also because the top-1 result usually means more to users in a search engine. The practical strategy used in the experiment is shown in Table 3, in which $sim(qr, R(qr))$ denotes the similarity between the query qr and the title of the top-1 search result $R(qr)$. The similarity is computed based on word overlap rate. $sim(qs, R(qs))$ is computed in the same way. Thresholds T_1 and T_2 were empirically set as 0.7 and 0.8, respectively.

Experimental results show that the top-1 search results for 65 (out of 765) test queries were changed using the above strategy, which means that our method has replaced the original search results of these queries with that of the generated questions. We asked the annotators to compare the search results before and after using the strategy. A query is annotated as *Good* if its new result is better than the old one, *Bad* if the new result is worse than the old one, and *Same* if the quality of the result is not evidently changed. Annotation result is shown in the first line of Table 4. As can be seen, the top-1 search results for 27

Input:	qr: user query
	qs: generated 1-best question for qr
	ZD: Zhidao cQA archive
Output:	$RST(qr)$: top-1 search result of qr in ZD
1. Search qr in ZD, get top-1 result $R(qr)$	
2. Search qs in ZD, get top-1 result $R(qs)$	
3. Compute similarity $sim(qr, R(qr))$	
4. Compute similarity $sim(qs, R(qs))$	
5. **If** $sim(qr, R(qr)) < T_1$ and $sim(qs, R(qs)) > T_2$	
6. Return $RST(qr) = R(qs)$	
7. **Else**	
8. Return $RST(qr) = R(qr)$	

Table 3: Strategy used in the experiments.

	Good	Same	Bad	Total
All que.	27	31	7	65
Correct que.	26	23	4	53
Incorrect que.	1	8	3	12

Table 4: Evaluating the effectiveness of query-to-question generation in cQA search.

queries get better, while that for only 7 queries get worse. This result demonstrates that our question generation technique can improve the performance of cQA search. Take the following case for example. The original query is " *读后感* (*book review*)"， whose top-1 search result is "*假如给我三天光明的读后感* (*book review of 'Three Days to See'*)". Our method can generate a question "*读后感怎么写* (*How to write a book review*)" for the query and accordingly retrieve this question as the top-1 result from Zhidao. It is obvious that the new result is much more likely to be looked for by users than the old one.

We also analyzed the search results that got worse after using our strategy. It is found that 6 of the 7 are still correct but not so good as the old ones. There is only one search result becomes unrelated to the query. Moreover, we should note that the 1-best questions used for improving cQA search are not necessarily correct. Hence we further evaluated the performance when only considering the correct or incorrect 1-best questions. The evaluation results are depicted in line 2 and 3 of Table 4. It is interesting to find that the performance did not evidently decrease when we only used the incorrect 1-best questions in our strategy. This result indicates that the performance of the query-to-question generation module in cQA search is not sensitive to the generation errors if we adopt a proper strategy to integrate it into the cQA site search engine.

4 Related Work

4.1 Research on cQA

Several studies have been carried out on cQA. Some of them have focused on retrieval and recommendation on cQA archives. For example, Xue et al. (2008) designed a retrieval model for cQA search, which considers both question and answer parts when measuring the relatedness between queries and cQA resources. Wang et al. (2009) presented a syntactic tree matching method for finding similar questions. Cao et al. (2008) proposed a question recommendation method based on tree cut model. Wang et al. (2010) proposed a graph-based approach to segmenting multi-sentence questions, so as to improve search performance. Another category of studies on cQA aims to estimate the quality of questions, answers, and users. For example, Song et al. (2008) examined the utility of questions in the cQA archives. Liu et al. (2008) presented a classification-based method to automatically predict whether a question-asker will be satisfied with the answers. Jurczyk and Agichtein (2007) tried to identify experts in a cQA community.

4.2 Question Generation

Question generation is a branch of natural language generation, which is defined as the task of automatically generating questions from some form of input (Rus and Graesser, 2009). The input may vary from a deep semantic representation to a raw text. Previous studies on question generation have mostly focused on text-to-question generation, which generates questions from declarative sentences or paragraphs. This technique is useful in education, especially in reading tutoring. Most previous studies employed rule-based methods in their text-to-question generation systems (Ali et al., 2010; Kalady et al., 2010; Mannem et al., 2010; Pal et al., 2010; Piwek and Stoyanchev, 2010; Varga and Ha, 2010).

Query-to-question generation is a sub-task of question generation, which was first proposed by Lin (2008). Lin has suggested to learn query-to-question generation models with query logs. However, no detail method or evaluation has been presented. There has been no other research since,

either, which may be mainly because few researchers can access the query log data.

4.3 Query Reformulation

Query reformulation is an important topic in the IR community, since it can improve users' search experience. Query reformulation mainly involves query reduction, expansion, and spelling correction. Query-to-question generation is closely related to query expansion. However, its goal is not only expanding useful information for the original query, but also organizing the information to produce a question, with which one can better understand the query's structure and the user's intent.

Several techniques have been proposed for query reformulation, which are mostly based on relevance feedback (Xu and Croft, 1996; Mitra et al., 1998) and query log analysis. Especially, the studies based on query logs can be divided into three categories. In the first one, researchers learn related query pairs from query sessions. The basic idea is that queries from the same session are more likely to be related to each other (Fonseca et al., 2005; Jones et al., 2006; Zhang and Nasraoui, 2006). The second kind of method identifies related queries using click-through information. They assume that queries leading to similar clicks are related in meaning (Wen et al., 2002; Baeza-Yates and Tiberi, 2007). The third category of method directly learns expansion terms from the clicked documents of the query. Their hypothesis is that terms in a query and a user-clicked document might be related (Cui et al., 2002; Riezler et al., 2008).

4.4 Template Induction

Template induction has been widely researched in NLP community, in which the following studies are close to our work. (1) Answer template learning in QA. Some QA systems use templates in answer extraction, which can be learned from large Web corpora or Web search results with handcrafted seed tuples (Ravichandran and Hovy, 2002). (2) Query template acquisition. For example, Agarwal et al. (2010) mine templates from search engine query logs with the goal of query interpretation. Szpektor et al. (2011) extract templates for long-tail queries so as to improve query recommendation. (3) Paraphrase template learning. Paraphrase templates are templates that can convey identical information when the variables are instantiated with the same contents. Paraphrase templates can be learned from monolingual corpora based on distributional hypothesis (Lin and Pantel, 2001), from comparable news articles based on alignment (Barzilay and Lee, 2003), or from bilingual corpora based on pivot approaches (Zhao et al., 2008). The above-mentioned studies are all related to our work. However, none of the previous work addresses the problem of query-to-question template generation.

5 Conclusions and Future Work

This paper addresses the problem of query-to-question generation for cQA and proposes a method based on search engine query logs. Several conclusions can be drawn from the experimental results. First, search engine query logs are powerful data for the research of query-to-question generation, from which we have acquired a large volume of question generation templates. Second, the proposed method is effective, which achieves promising precision and outperforms a baseline method. Third, the query-to-question generation technique can be used to improve the search of cQA archives.

In our future work, we will exploit larger-scale query logs for acquiring question generation templates. We will also improve the similar query retrieval strategy (Section 2.2), which underperforms now on single-word queries. In addition, we will have a deeper insight into the applications of question-to-query generation in cQA services.

Acknowledgments

This work was supported by (1) China Postdoctoral Science Foundation (No.20100480100), and (2) Beijing Postdoctoral Research Foundation.

References

Ganesh Agarwal, Govind Kabra, and Kevin Chen-Chuan Chang. 2010. Towards Rich Query Interpretation: Walking Back and Forth for Mining Query Templates. In *Proceedings of WWW*, pages 1-10.

Husam Ali, Yllias Chali, and Sadid A. Hasan. 2010. Automation of Question Generation From Sentences. In *Proceedings of the Third Workshop on Question Generation*, pages 58-67.

Ricardo Baeza-Yates and Alessandro Tiberi. 2007. Extracting Semantic Relations from Query Logs. In *Proceedings of KDD*, pages 76-85.

Regina Barzilay and Lillian Lee. 2003. Learning to Paraphrase: An Unsupervised Approach Using

Multiple-Sequence Alignment. In *Proceedings of HLT-NAACL*, pages 16-23.

Yunbo Cao, Huizhong Duan, Chin-Yew Lin, Yong Yu, and Hsiao-Wuen Hon. 2008. Recommending Questions Using the MDL-based Tree Cut Model. In *Proceedings of WWW*, pages 81-90.

Hang Cui, Ji-Rong Wen, Jian-Yun Nie, Wei-Ying Ma. 2002. Probabilistic Query Expansion Using Query Logs. In *Proceedings of WWW*, pages 325-332.

Bruno M. Fonseca, Paulo Golgher, Bruno Pôssas, Berthier Ribeiro-Neto, Nivio Ziviani. 2005. Concept-based Interactive Query Expansion. In *Proceedings of CIKM*, pages 696-703.

Zellig Harris. 1985. Distributional Structure. In *The Philosophy of Linguistics*, pages 26-47.

Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. 2006. Generating Query Substitutions. In *Proceedings of WWW*, pages 387-396.

Pawel Jurczyk and Eugene Agichtein. 2007. Hits on Question Answer Portals: Exploration of Link Analysis for Author Ranking. In *Proceedings of SIGIR*, pages 845-846.

Saidalavi Kalady, Ajeesh Elikkottil, Rajarshi Das. 2010. Natural Language Question Generation Using Syntax and Keywords. In *Proceedings of the Third Workshop on Question Generation*, pages 1-10.

J. R. Landis and G. G. Koch. 1977. The Measurement of Observer Agreement for Categorical Data. In *Biometrics* 33(1): 159-174.

Chin-Yew Lin. 2008. Automatic Question Generation from Queries. In *Proceedings of Workshop on the Question Generation Shared Task and Evaluation Challenge*.

De-Kang Lin and Patrick Pantel. 2001. Discovery of Inference Rules for Question Answering. In *Natural Language Engineering* 7(4): 343-360.

Yandong Liu, Jiang Bian and Eugene Agichtein. 2008. Predicting Information Seeker Satisfaction in Community Question Answering. In *Proceedings of SIGIR*, pages 483-490.

Prashanth Mannem, Rashmi Prasad, and Aravind Joshi. 2010. Question Generation from Paragraphs at UPenn: QGSTEC System Description. In *Proceedings of the Third Workshop on Question Generation*, pages 84-91.

Mandar Mitra, Amit Singhal, and Chris Buckley. 1998. Improving Automatic Query Expansion. In *Proceedings of SIGIR*, pages 206-214.

Santanu Pal, Tapabrata Mondal, Partha Pakray, Dipankar Das and Sivaji Bandyopadhyay. 2010. QGSTEC System Description - JUQGG: A Rule based approach. In *Proceedings of the Third Workshop on Question Generation*, pages 76-79.

Paul Piwek and Svetlana Stoyanchev. 2010. Question Generation in the CODA project. In *Proceedings of the Third Workshop on Question Generation*, pages 29-34.

Deepak Ravichandran and Eduard Hovy. 2002. Learning Surface Text Patterns for a Question Answering System. In *Proceedings of ACL*, pages 41-47.

Stefan Riezler, Yi Liu, and Alexander Vasserman. 2008. Translating Queries into Snippets for Improved Query Expansion. In *Proceedings of COLING*, pages 737-744.

Vasile Rus and Arthur C. Graesser. 2009. Workshop Report: The Question Generation Task and Evaluation Challenge. Institute for Intelligent Systems, Memphis, TN, ISBN: 978-0-615-27428-7.

Idan Szpektor, Aristides Gionis, and Yoelle Maarek. 2011. Improving Recommendation for Long-tail Queries via Templates. In *Proceedings of WWW*, pages 47-56.

Young-In Song, Chin-Yew Lin, Yunbo Cao and Hae-Chang Rim. 2008. Question Utility: A Novel Static Ranking of Question Search. In *Proceedings of AAAI*, pages 1231-1236.

Andrea Varga and Le An Ha. 2010. WLV: A Question Generation System for the QGSTEC 2010 Task B. In *Proceedings of the Third Workshop on Question Generation*, pages 80-83.

Kai Wang, Zhaoyan Ming, Tat-Seng Chua. 2009. A Syntactic Tree Matching Approach to Finding Similar Questions in Community-based QA Services. In *Proceedings of SIGIR*, pages 187-194.

Kai Wang, Zhaoyan Ming, Xia Hu, Tat-Seng Chua. 2010. Segmentation of Multi-Sentence Questions: Towards Effective Question Retrieval in cQA Services. In *Proceedings of SIGIR*, pages 387-394.

Ji-Rong Wen, Jian-Yun Nie, and Hong-Jiang Zhang. 2002. Query Clustering Using User Logs. In *ACM Transactions on Information Systems* 20(1): 59-81, 2002.

Jinxi Xu and W. Bruce Croft. 1996. Query Expansion using Local and Global Document Analysis. In *Proceedings of SIGIR*, pages 4-11.

Xiaobing Xue, Jiwoon Jeon, W. Bruce Croft. 2008. Retrieval Models for Question and Answer Archives. In *Proceedings of SIGIR*, pages 475-482.

Zhiyong Zhang and Olfa Nasraoui. 2006. Mining Search Engine Query Logs for Query Recommendation. In *Proceedings of WWW*, pages 1039-1040.

Shiqi Zhao, Haifeng Wang, Ting Liu, and Sheng Li. 2008. Pivot Approach for Extracting Paraphrase Patterns from Bilingual Corpora. In *Proceedings of ACL-08:HLT*, pages 780-788.