Ranking Methods for Query Relaxation in Book Search

Momo Kyozuka
Graduate School of Informatics, Kyoto University
Kyoto, Japan
kyozuka@dl.soc.i.kyoto-u.ac.jp

Keishi Tajima
Graduate School of Informatics, Kyoto University
Kyoto, Japan
tajima@i.kyoto-u.ac.jp

***Abstract*—This paper proposes a method to support book search tasks where users only have vague memories of the story or the contents of the books. Queries produced by users with such vague memories may include extraneous or even wrong words, and may not match with the descriptions of the books in the database. We need some query relaxation scheme to find the book with such erroneous queries including excessive words. To develop such a scheme, we first analyze what kind of words are more likely to be extraneous or wrong in book descriptions written by users with vague memories. We classify words into four types based on their roles in the description, and estimate the probability of their appearance in the database description for each type. When given a description of a book by a user, we generate queries by using every subset of the words in the description, and rank the queries based on expected ranking of the target book in their results. Expected ranking of the target book is estimated by using appearance probabilities of words in queries and the number of books matching the queries. We conducted an experiment for comparing various ranking schemes and evaluated their performance by MRR. The result shows that the ranking scheme that use both the word appearance probabilities and the number of matching books outperforms the other schemes.**

***Index Terms*—query relaxation, query correction, query ranking, query suggestion, query recommendation**

1. Introduction

Most libraries have a service called “reference service”, which helps users finding books they are looking for. Sometimes users are looking for a book they read before but do not remember its title and authors. In that case, the users describe the contents of the book based on their memory, and librarians try to find books matching the description.

Some libraries archive questions and answers at their reference services in the past into a database. For example, Collaborative Reference Database[1] is a database managed by National Diet Library of Japan, which archives questions and answers collected from many libraries in Japan. Many questions in this archive are very vague descriptions of the stories of books the users read long before, e.g., in their childhood. Similar type of questions are also often found in QA sites, such as Yahoo! Chiebukuro[2], which is the most popular QA site in Japan.

Libraries also have a database storing brief descriptions of contents of the books published in the past, and librarians use the database to identify the books users are looking for. However, if we simply use the descriptions by the users as queries on the database, the queries very often do not match with the correct book. It is because the descriptions are based on very vague memories of the users and often include wrong keywords, which does not appear in the description of the target book in the database. The description by the users also often include extraneous words that are not wrong but does not happen to appear in the description in the database.

In this paper, we propose a method to support retrieval of books by queries produced from a description based on vague memories of users. Because such queries often include many words that do not appear in the descriptions of the target books in the database, we often need to remove some words from the given query. In other words, we need some query relaxation scheme that is appropriate in this scenario. In this paper, we propose a method that generates relaxed queries by using every subset of the words in the description by the user, and rank the generated queries.

The outline of our method of ranking generated queries is as follows. First, each word in the description by the user is classified into the following four types based on their roles in the sentences: subject, predicate, object, and others. For each type, we estimate the probability that a word of that type in a user description is correct and appears in the description of the target book in the database. We estimated it based on the statistics we obtained from the archive of a QA site. By using these probabilities for each word, we calculate the probability that each generated query matches with the description of the target book in the database.

In addition to the probability of matching with the correct book, we also count the number of matching books in the database for each query. By using these two, we calculate the expected rank of the target book in the result list of each query. We rank queries based on this value, and we concatenate the result lists of all queries in that order for

1. http://crd.ndl.go.jp/reference/

2. https://chiebukuro.yahoo.co.jp

producing the final result shown to the user. In this method, even if a query has high probability of matching with the target book, if it has a huge number of matching books, the query may be ranked lower than another query that has lower probability but has a smaller number of matching books.

In order to evaluate our method, we collected 50 pairs of a question and an answer from a QA site in Japan where the question describes a story of a book and the answer includes the title of a book which is confirmed by the user as the book she was looking for. We then generated queries by using the description in the questions, submitted the queries to the book search system[3] provided by National Diet Library in Japan, and compared the result with several baseline methods. The result of our experiment shows that our scheme outperforms other baseline methods.

2. Related Work

There have been much research on query recommendation [1], which is also called query suggestion [2]–[4]. Query suggestion can be classified into several types. The most general type of query suggestion would suggest any queries as long as they are chosen by the suggestion method.

On the other hand, there have been much research on suggestion of additional keywords to the given query, which is called query completion. (It is also sometimes called query expansion [5], [6], but the name query expansion is also used to refer to more general type of query suggestion [7].) Some Web search engines show candidates of the next keyword when users type query keywords into the query box. It is an example of query completion [8], [9].

On the other hand, query suggestion removing existing words from given queries are sometimes called query relaxation. There have been much research on query completion, which add words to given queries, but there have not been much research on query relaxation for keyword queries (while there have been much research on query relaxation for structural queries [10], [11]). It is probably because it is more difficult for users to come up with additional keywords than to remove some existing keywords.

Query relaxation is, however, useful when users need to improve the recall. Query relaxation removes some keywords or replace some keywords with less restrictive words, such as their hypernyms. It may also replace some keyword with a disjunction of all its synonyms. Muslea [12] proposed a method of finding relaxed non-empty queries that are closest to a given query that returns empty result.

Our method is also a kind of query relaxation. We focus on tasks of finding books based on users' vague memory of the contents, and analyzed what kind of words are likely to be missing in the descriptions in the database. We also propose a novel idea of ranking relaxed queries based on the expected rank of the correct answer in their results.

Kaneko et al. [13] also proposed a method of query relaxation. In their method, users can specify the allowance degree of the replacement of each word in the query. In this paper, we assume that users do not know which words may be wrong, but if that information is available, it must be very useful for improving the ranking of the relaxed queries. It is an interesting direction for extending our method.

Many existing methods for query suggestion use the information on the choices made by users in the past that are recorded in query logs [1]–[4], [14]. However, we focus on queries including wrong words because of the vague memories, and the probability that queries including the exactly same error exist in the query log is not high. Instead of using the information on the same queries in query logs, we assume that the probability of errors have correlation with the type of words, and we obtain statistics on that from the archive of questions and answers in the past.

In this paper, we focus on queries produced from a description of the contents of a book, which usually include many words. Queries with too many words often cannot produce appropriate results compared with shorter queries [15]. There have been research on solving the problem of "long queries" by eliminating extra words and generating shorter queries that preserves query intents in the original query as much as possible. Chen and Zhang [16] proposed a method for producing shorter queries based on query log information. We also use the information in the past questions and answers to rank shorter relaxed queries.

Kumaran and Carvalho [17] and also Balasubramanian, et al. [18] proposed a method of producing short queries from long queries by using various estimators of query quality and learning-to-rank approach with RankSVM [19]. Our method uses the probability that words appear in the document, which is also a kind of estimator of query quality.

Ochiai et al. [20] conducted an experiment on human episode memory. They ask users to read news articles, and later ask them to specify queries for retrieving the articles. The result shows that the query specified by the users long after they have read articles include more verbs, and it degrades the performance of the queries. This result agrees with our observation that predicates are more likely to be wrong than subjects and objects.

Kim et al. [21] proposed a method of query suggestion for academic paper search tasks. Their method extracts what they call phrasal-concepts, which are subject-specific phrases used for describing ideas in academic papers. On the other hand, our method extracts phrases representing the relationship between two query terms in general documents.

3. Proposed Method

In this section, we explain the details of our method.

3.1. Word Classification

In order to obtain statistics on what type of words in book descriptions by users are likely to be correct and appear in the descriptions in the database, we collected 50 pairs of questions and answers where the correct book was identified. We collected them from Yahoo! Chiebukuro, a popular QA

3. http://iss.ndl.go.jp

TABLE 1. THE RATIO OF WORDS APPEARING IN THE AMAZON DATA

class	in questions	in database	ratio
Subject	48	30	0.625
Predicate	73	12	0.164
Object	65	37	0.569
Others	41	18	0.439

TABLE 2. THE RATIO OF WORDS APPEARING IN THE NDL DATA

class	in questions	in database	ratio
Subject	43	19	0.442
Predicate	62	3	0.048
Object	55	30	0.545
Others	34	15	0.441

site in Japan. We also collected the description of these books from Amazon.co.jp[4] (Amazon) and also from the web site of National Diet Library of Japan (NDL).

We then extract the main sentence in the description of the target book written by the user, and classify words in the sentence into the following four types based on the role that each word plays in the sentence:

- subjects,
- predicates,
- objects, and
- others.

Predicate is the concept used in Japanese grammar, and it roughly corresponds to verbs and predicative adjectives.

We then examined whether each word also appears in the description of the corresponding book obtained from Amazon and NDL, and calculated the ratio of the appearing words for each of the four types above. When calculating it with data from NDL, we excluded 9 books for which we cannot find description data in NDL.

Table 1 and Table 2 show the result with the data obtained from Amazon and NDL, respectively. In the descriptions from Amazon, subject appears most frequently, and objects, others and predicates follows in this order. In the descriptions from NDL, objects appears most frequently, and subject, others, and predicates follows in this order. We use these ratio as the approximation of the probability of appearance of each type of words in the description in the database.

3.2. Query Ranking Methods

Given a description of a book written by a user, we extract the main sentence, and extract nouns, verbs, and adjectives in it. Next, we produce keyword queries by using every subset of these words, and rank them. Finally, the result lists of these queries are concatenated in the order of the ranking of the queries, and the concatenated list is shown to the user as the final result of their query.

For ranking the generated queries, we compared the following 12 ranking methods. The last two are the methods we propose in this paper, and we compare their performance with the other 10 methods in the experiment.

Random Repeated Removal

In this method, we produce a ranked list of queries by starting from the original query, and removing a randomly chosen word one by one. In this and the next method, we produce only $n-1$ queries from the original query including n words, while we create 2^n-1 queries in the other methods.

Priority-based Repeated Removal

In this method, we produced a ranked list of queries by starting from the original query, and removing a word with the lowest probability of the appearance one by one, i.e., in the order of predicates, others, objects, and subjects. For example, if the query includes a predicate and two "others" words, the predicate is removed first, then one of the "others" are randomly chosen and removed, and finally the remaining "others" is removed.

Random Ranking

We rank all $2^n - 1$ queries randomly.

Number of Words

In this method, we rank the queries by the number of included words in the descending order. Queries with the same number of words are ranked randomly.

Number of Words and Priority

In this method, we rank the queries by the number of included words in the descending order, and rank the queries with the same number of words by the appearance probability of removed words in the ascending order.

Sum of Probability

We rank queries by the sum of the appearance probability of words in the query in the descending order. For each type of words, we use the average of appearance probability for Amazon data and NDL data, i.e., subjects: 0.5335, predicates: 0.106, objects: 0.557, and others: 0.44.

Number of Words and Sum of Probability

We first rank queries by the number of included words in the descending order, and rank those with the same number of words by the sum of the probability explained above.

Probability Ranking

Let $P(q)$ denote the probability that all included words in the query q appear in the description in the database. We rank queries by $P(q)$ in the descending order. In other words, we rank queries by the product of probabilities of words in the queries. In this method, queries with a smaller number of words tend to be ranked higher.

Number of Words and Probability

We first rank queries by the number of included words in the descending order, and rank those with the same number of words by $P(q)$ explained above.

TF-IDF

For each word a included in the original query, we define its *tfidf* value as follows:

$$tfidf(a) = tf(a) \times \log \frac{N}{df(a)}$$

where $tf(a)$ is 1 when the word a is included in a generated query, and 0 when it is not included, $df(a)$ is the number of books in the NDL database matching to the query a, and

4. https://www.amazon.co.jp

TABLE 3. EXAMPLES OF WORD CLASSIFICATION

book title	description	word classification
The Dwarfs of the Tree House	A girl is friendly with dwarfs	(subject: a girl) (object: dwarfs) (predicate: is friendly)
The Scary and Dangerous Excursion	A girl gets stuck in a fairy tale world	(subject: a girl) (others: a fairy tale) (object: world) (predicate: gets stuck)

N is the sum of $df(x)$ for all words extracted from all the question data.

In this method, we create a *tfidf* vector whose elements are $tfidf(a)$ for each original query and each generated queries, and calculate the cosine similarity between the vector of the original query and the vector of the generated query. We rank the generated queries by the similarity to the original query in the descending order.

Expected Rank of Target

Let $hit(q)$ denote the number of books matching the query q. We calculate the expected rank of the target book in the result list of q, denoted by $\hat{r}(q)$, by the formula below:

$$\hat{r}(q) = \frac{1}{2}P(q)hit(q) + (1 - P(q))(hit(q) + 1)$$

In other words, we assume that the target book appears in the random rank in the query result in the probability $P(q)$, and appears at the rank $hit(q) + 1$ (i.e., appear as the item next to the last item in the result list of q, which is produced by the query ranked next to q) in the probability $1 - P(q)$.

We rank the queries by $\hat{r}(q)$ in the ascending order.

Number of Words and Expected Rank of Target

We first rank queries by the number of included words in the descending order, and rank those with the same number of words by $\hat{r}(q)$ explained above.

4. Experiments

In this section, we explain the experiment we conducted for comparing the ranking methods. To create a dataset, we collected 50 question-answer pairs from Yahoo! Chiebukuro as explained before, and then removed those whose book do not have the description data in NDL database. We also removed cases where no words in the description in the question matches with the description of the target book in the database. On the other hand, sometimes the target book in the question has multiple database entries in NDL database with different description data. In such cases, we include all the entries in our dataset. As the result, we obtained data of 27 questions and 37 books.

4.1. Generating and Ranking Queries

Table 3 shows two examples of the main sentences extracted from the descriptions by users. One of them is looking for a book whose title is: The Dwarfs of the Tree House (in Japanese: Kokage No Ie No Kobito Tachi), and the other is looking for a book whose title is: The Scary and Dangerous Excursion (in Japanese: Ensoku Kowaizo Abunaizo). Table 3 also shows the classification of words in the sentences.

TABLE 4. RANKING OF QUERIES BY EXPECTED RANK OF TARGET (PROPOSED METHOD 1)

Predicate	Others	Object	Subject
gets stuck	a fairy tale		
gets stuck	a fairy tale	world	
gets stuck	a fairy tale		a girl
gets stuck	a fairy tale	world	a girl
gets stuck		world	a girl
gets stuck			a girl

TABLE 5. RANKING OF QUERIES BY NUMBER OF WORDS AND EXPECTED RANK OF TARGET (PROPOSED METHOD 2)

Predicate	Others	Object	Subject
gets stuck	a fairy tale	world	a girl
gets stuck	a fairy tale	world	
gets stuck	a fairy tale		a girl
gets stuck		world	a girl
gets stuck	a fairy tale	world	
gets stuck	a fairy tale		

Table 4 and Table 5 show examples of a ranking of queries produced by the proposed methods. Table 6 to Table 11 also shows examples of query rankings produced by some of the methods explained in the previous section. (In Table 4 to Table 11, only the top 6 queries are shown.)

4.2. Executing Queries and Concatenating Results

We executed each generated queries on NDL database in their ranking order, and recorded the number of books in their search results. We also examined which query is the first query that includes the target book in its result, and recorded the rank of the book in that result.

The multiple keyword search on NDL shows a list of books whose description data include all the query keywords. They are listed in the order of relevance uniquely defined in NDL search system[5]. When there are more than 500 matching books, they only show the top 500 books as the search result.

If the description of the target book includes all the query keywords in the i-th query q_i and so the book must be included in the entire search result of the query, but is not shown in the top 500, we approximate the rank of the target book by the rank just at the middle between 500th and the end of the list. Therefore, we define the expected rank of the target book in such a case by the following formula:

$$\hat{r}(q_i) = \frac{1}{2}(500 + hit_i)$$

where hit_i is the number of books matching with q_i.

If the target book first matches with the query ranked at the k-th position in the query ranking, $rank$, which denotes

5. http://iss.ndl.go.jp/information/faq/#B2

TABLE 6. RANKING OF QUERIES BY PRIORITY-BASED REPEATED REMOVAL

Predicate	Others	Object	Subject
gets stuck	a fairy tale	world	a girl
	a fairy tale	world	a girl
		world	a girl
			a girl

TABLE 7. RANKING OF QUERIES BY NUMBER OF WORDS AND PRIORITY

Predicate	Others	Object	Subject
gets stuck	a fairy tale	world	a girl
	a fairy tale	world	a girl
gets stuck	a fairy tale		a girl
gets stuck	a fairy tale	world	
		world	a girl
	a fairy tale		a girl

TABLE 8. RANKING OF QUERIES BY SUM OF PROBABILITY

Predicate	Others	Object	Subject
gets stuck	a fairy tale	world	a girl
	a fairy tale	world	a girl
gets stuck		world	a girl
gets stuck	a fairy tale	world	
		world	a girl
gets stuck	a fairy tale		a girl

TABLE 9. RANKING OF QUERIES BY NUMBER OF WORDS AND SUM OF PROBABILITY

Predicate	Others	Object	Subject
gets stuck	a fairy tale	world	a girl
	a fairy tale	world	a girl
gets stuck		world	a girl
gets stuck	a fairy tale	world	
gets stuck	a fairy tale		a girl
		world	a girl

TABLE 10. RANKING OF QUERIES BY PROBABILITY

Predicate	Others	Object	Subject
		world	
		world	a girl
	a fairy tale	world	
			a girl
	a fairy tale		
	a fairy tale	world	a girl

TABLE 11. RANKING OF QUERIES BY NUMBER OF WORDS AND PROBABILITY

Predicate	Others	Object	Subject
gets stuck	a fairy tale	world	a girl
	a fairy tale	world	a girl
gets stuck		world	a girl
gets stuck	a fairy tale	world	
gets stuck	a fairy tale		a girl
		world	a girl

the rank of the target book in the final result is defined as follows:

$$rank = \sum_{i=1}^{k-1} hit_i + \hat{r}(q_k)$$

where hit_i is the number of books matching with the i-th query and $\hat{r}(q_k)$ is the rank of the target book in the search result of the k-th query. If the target book is not included in the search results of all the generated queries, the rank of the target book in the final result is ∞.

If there are multiple data entry with different description data for the same book in NDL database, we calculated the rank of each data entry and let the smallest one be the rank of the target book.

TABLE 12. COMPARISON BY MRR

Method	MRR
Random Repeated Removal (average)	0.004938272
Simple Repeated Removal	0.098947087
Random Ranking (average)	0.091820651
Random Ranking (worst)	0.000421639
Ranking by Number (average)	0.164337041
Ranking by Number (worst)	0.104480127
Number and Priority	0.152101894
Sum of Probability	0.148613002
Number and Sum of Probability	0.148610991
Probability Ranking	0.044323034
Number and Probability	0.149009630
TF-IDF	0.120593996
Expected Rank of Target	0.156152319
Number of Words and Expected Rank of Target	0.139432412

4.3. Evaluation

We compared the 12 ranking methods explained in the previous section by Mean Reciprocal Rank (MRR), which is defined by the formula below.

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i}$$

$|Q|$ is the number of original questions, which is 27 in this experiment. $rank_i$ is the $rank$, which is defined before, for the i-th original query. If the target book matches with no query generated for the i-th question, $\frac{1}{rank_i}$ is defined as 0.

Among 12 methods, the Random Repeated Removal method and the Priority-Based Repeated Removal methods produce only $n-1$ queries among possible 2^n-1 queries. For these two, we were also interested in the following question: for how many questions out of 27 questions could they produce at least one query that matches the target book? We investigated the answer to this question as well.

For the other 9 methods, we are interested not only MRR, which is the average performance over all queries, but also the variance of their performance, in other words, their robustness. In order to see the variance, we created histograms where horizontal axis represents the rank of the target book in the final merged query result, and the vertical axis represents the occurrence frequency ratio of the ranks.

Notice that some methods include random factors. In order to obtain the average performance of those methods, we took the average of 5 runs when calculating MRR of the Random Repeated Removal method and the Random Ranking method, and took the average of 10 runs when calculating the Ranking by Number method.

4.4. Result

The MRR of each method is shown in Table 12, and the histograms showing the variance of the ranking of the target book are shown in Figure 5 to Figure 10.

First, we compare Random Removal and Priority-Based Removal methods. The latter achieved 20 times better MRR than the former. The latter could produce queries matching with the target book in 10 cases out of 27 questions. On the other hand, we run the former method 5 times for each question, and in 2 runs out of 5, the former method could produce queries matching with the target book only in one case out of 27 questions, and in 3 runs out of 5, it could produce matching queries in no case out of 27

questions. These results show that the method that removes query keywords based on the statistics shown in Table 1 and Table 2 shows far better performance than the random method.

Next, we compare the other 10 methods by MRR. One of our proposed method Expected Rank of Target achieved the second highest MRR among them, but the Ranking by Number of Words method achieved a slightly better MRR in average. However, the Ranking by Number of Words method has larger variance of MRR than our method as shown in Figure 3 and Figure 11. In other words, our method is only slightly worse than the Ranking by Number of Words method in average, but is more stable. In fact, the MRR of that method in the worst case is far lower than that of our method as shown in Table 12.

Based on these experimental results, we will discuss advantages and disadvantages of some method, and also the future direction for the improvement of some methods.

4.5. Priority-Based Repeated Removal

The Priority-Based Repeated Removal method achieved far better MRR than the Random Removal method. It is mainly because the method could produce queries matching the target book more often than the Random Removal.

However, this method still failed to produce queries matching the target book for about two thirds of questions. When the original query contains multiple words of the same class, those words were randomly ranked in this method. We could improve this method by using more appropriate strategy at this step, e.g., considering the order of those words in the original query.

The inherent shortcoming of this method is that it cannot produce queries where some words with higher probability (i.e., lower priority in the removal process) are removed while some words with lower probability are kept. The result of our experiment proved that such queries are sometimes useful, and the methods that do not produce such queries at all are insufficient.

4.6. Priority Ranking; TF-IDF

MRR of the Number of Words and Priority method was next to our proposed method. However, the variance of MRR of this method was large as shown in Figure 5. In this method, the rank of the target book in the final result sometimes exceeded 80000. The TF-IDF method also showed a similar result.

4.7. Sum of Probability; Number and Sum of Probability

The MRR of these two methods were also close to that of the best methods. In these methods, queries with many words tend to be ranked higher. Therefore, the number of books matching with the highly ranked queries, which include many words, tend to be small. As a result, if some of those highly ranked queries matches with the target book, the rank of the target book is high, and even if some highly ranked query fail to match with the target book, it does not drastically lower the rank of the target book as long as it matches with some of the immediately following queries.

In other words, these ranking methods have a effect that is very similar to that of our proposed method that uses the expected rank of the target book.

4.8. Probability Ranking

The MRR of the Probability Ranking method was very low, even lower than the average of the Random Ranking method. It is because this method tend to rank queries with many words lower, because the product of probabilities of all words become lower when we have many words.

As a result, it has the effect opposite to that of the Sum of Probability method or the Number and Sum of Probability method. Queries with a few words, which usually have many matching books, are ranked higher, and if their results do not include the target book, the rank of the target book becomes significantly low even if it matches with the next-ranked query.

However, the histograms show that the rank of the target book in the worst case of this method is still higher than 60,000, and it is better than many other methods. It is because the target book matches with some of the top-ranked queries in most cases in this method. However, the rank of the target book is often low simply because those queries have a large number of matching books, and the target book often appear in the lower half of them.

4.9. Number of Words and Probability

The MRR of this method is significantly higher than that of Probability Ranking. It is because we can avoid the problem explained above by giving priority to queries with many words.

However, if the description of the target book in the database has only a few words included in the question, all queries matching with the target book has only a few words, and are never ranked high. For such books, this method is not appropriate. This is a drawback common to all methods that primarily rank queries by using the number of words.

4.10. Expected Rank of Target; Number of Words and Expected Rank of Target

The MRR of the former method was the second highest as mentioned before. When a query has many words, the probability that the description in the database includes all of them is low, but if it matches the target book, such queries with many words can rank the target book at very high rank. This method can control this trade-off more appropriately than the other methods by calculating the expected rank of the target book. The result of the experiment shows that our strategy aiming that effect really works.

The MRR of the latter method was lower than the former. In this method, we first rank queries by the number of included words in the descending order. The query with many words which does not have the target book in its search result is ranked higher than the query with less words which has the target book in its search result. This makes the target book ranked lower. It is the reason why the MRR of the latter method was lower.

There are several future directions to improve the approximation of the expected rank of the target book. For example, our current method does not consider the relationship between queries at all. A query can have strong correlation with other queries, and we can sometimes expect that the target book not matching with a query q_1 is unlikely to match q_2. We may be able to improve the approximation of the expected rank of the target book by using such a relationship between queries.

In addition, many queries have overlapping results, and when we concatenate the result of such queries, we eliminate the duplicated appearance of the same books. This duplicate elimination also influence on the expected rank of the target book. We should also include this factor in the computation.

5. Conclusion

In this paper, we proposed a method of helping users searching for books based on vague memories of the contents of the book.

First, we collected pairs of questions and answers related to book search tasks, and obtained statistics on what types of words that appear in the description by the users are more or less likely to be correct and to appear also in the description of the book in the database. We classified words into four types based on their role in the sentences, and obtained the statistics for each of them.

We then developed a method of ranking relaxed queries generated from an original query that may include extraneous words. We also proposed an idea of concatenating the query results of those queries in the order of their ranking, and showing the concatenated list to the user as the final result.

We also developed a query ranking scheme which is appropriate for this scheme of showing the final result to users. We estimate the expected rank of the target book in the final concatenated list, and rank the queries so that we can minimize the expected rank of the target book. Our method estimates the expected rank of the target book in the result of a given query based on the probability that the result of the query includes the target book, and also the number of books matching that query.

In this paper, we focused on the book search task, but the proposed scheme of merging the results of relaxed queries, and the proposed method of ranking relaxed queries based on the expected rank of the target data can be applied to a variety of scenarios where we need to generate many queries from the original query.

References

[1] R. Baeza-Yates, C. Hurtado, and M. Mendoza, "Query recommendation using query logs in search engines," in *Proc. of EDBT*, 2004, pp. 588–596.

[2] S. Cucerzan and R. W. White, "Query suggestion based on user landing pages," in *Proc. of SIGIR*, 2007, pp. 875–876.

[3] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li, "Context-aware query suggestion by mining click-through and session data," in *Proc. of KDD*, 2008, pp. 875–883.

[4] M. Shokouhi, M. Sloan, P. N. Bennett, K. Collins-Thompson, and S. Sarkizova, "Query suggestion and data fusion in contextual disambiguation," in *Proc. of WWW*, 2015, pp. 971–980.

[5] E. N. Efthimiadis, "Query expansion." *ARIST*, vol. 31, pp. 121–87, 1996.

[6] C. Carpineto and G. Romano, "A survey of automatic query expansion in information retrieval," *ACM Comput. Surv.*, vol. 44, no. 1, pp. 1:1–1:50, 2012.

[7] H. K. Azad and A. Deepak, "Query expansion techniques for information retrieval: a survey," *CoRR*, vol. abs/1708.00247, 2017.

[8] S. Whiting and J. M. Jose, "Recent and robust query auto-completion," in *Pro. of WWW*, 2014, pp. 971–982.

[9] G. Di Santo, R. McCreadie, C. Macdonald, and I. Ounis, "Comparing approaches for query autocompletion," in *Proc. of SIGIR*, 2015, pp. 775–778.

[10] S. Amer-Yahia, L. V. S. Lakshmanan, and S. Pandit, "Flexpath: Flexible structure and full-text querying for xml," in *Proc. of SIGMOD*, 2004, pp. 83–94.

[11] M. Yahya, D. Barbosa, K. Berberich, Q. Wang, and G. Weikum, "Relationship queries on extended knowledge graphs," in *Proc. WSDM*, 2016, pp. 605–614.

[12] I. Muslea, "Machine learning for online query relaxation," in *Proc. KDD*, 2004, pp. 246–255.

[13] Y. Kaneko, S. Nakamura, H. Ohshima, and K. Tanaka, "Query relaxation based on users' unconfidences on query terms and web knowledge extraction," in *Proc. of ICADL*, 2008, pp. 71–81.

[14] X. Wang and C. Zhai, "Mining term association patterns from search logs for effective query reformulation," in *Proce. of CIKM*, 2008, pp. 479–488.

[15] M. Bendersky and W. B. Croft, "Analysis of long queries in a large scale search log," in *Proc. of WSCD*, 2009, pp. 8–14.

[16] Y. Chen and Y.-Q. Zhang, "A query substitution-search result refinement approach for long query web searches," in *Proc. of WI-IAT*, 2009, pp. 245–251.

[17] G. Kumaran and V. R. Carvalho, "Reducing long queries using query quality predictors," in *Proc. of SIGIR*, 2009, pp. 564–571.

[18] N. Balasubramanian, G. Kumaran, and V. R. Carvalho, "Exploring reductions for long web queries," in *Proc. of SIGIR*, 2010, pp. 571–578.

[19] T. Joachims, "Optimizing search engines using clickthrough data," in *Proc. of KDD*, 2002, pp. 133–142.

[20] S. Ochiai, M. P. Kato, and K. Tanaka, "Re-call and re-cognition in episode re-retrieval: A user study on news re-finding a fortnight later," in *Proc. of CIKM*, 2014, pp. 579–588.

[21] Y. Kim, J. Seo, W. B. Croft, and D. A. Smith, "Automatic suggestion of phrasal-concept queries for literature search," *Information Processing & Management*, vol. 50, no. 4, pp. 568–583, 2014.

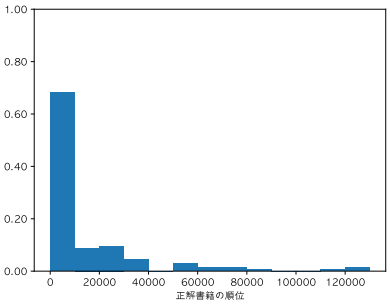

Figure 1. Random Ranking (5 runs total)

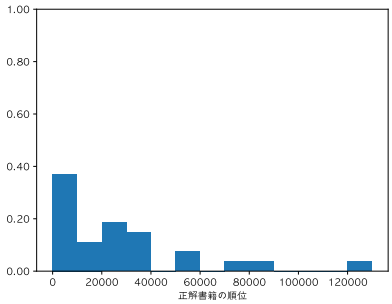

Figure 2. Random Ranking (worst)

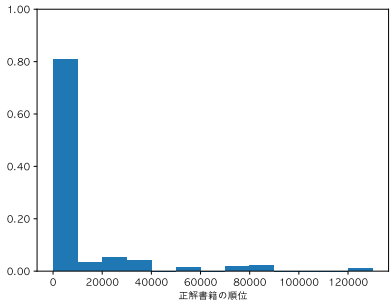

Figure 3. Ranking by Number (10 runs total)

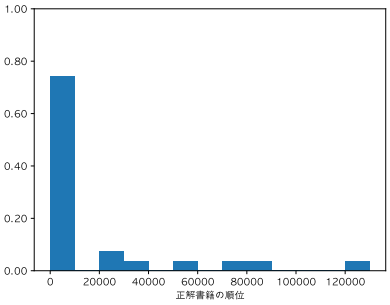

Figure 4. Ranking by Number (worst)

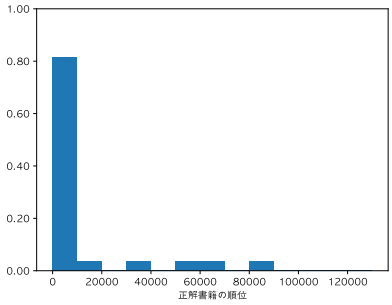

Figure 5. Ranking by Number and Priority

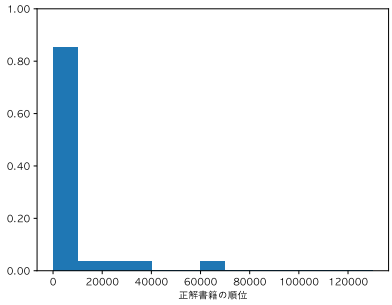

Figure 6. Ranking by Sum of Probability

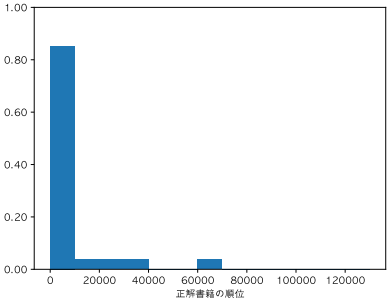

Figure 7. Ranking by Number and Sum of Prob.

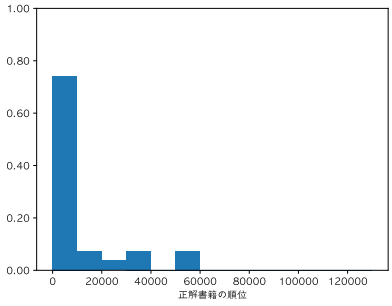

Figure 8. Probability Ranking

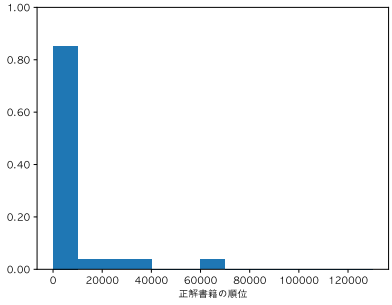

Figure 9. Ranking by Number and Probability

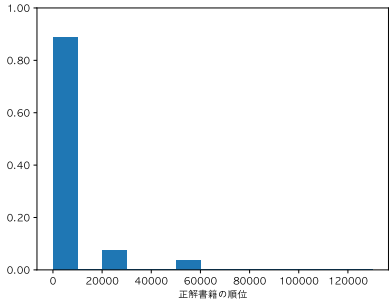

Figure 10. TF-IDF

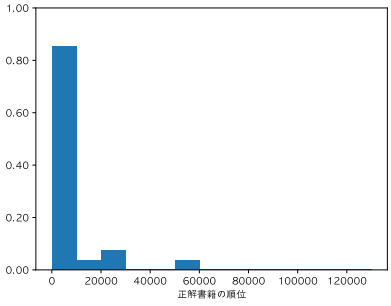

Figure 11. Expected Rank of Target