A Ranking Method for Relaxed Queries in Book Search

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

Yang Xu
xuyang@dl.soc.i.kyoto-u.ac.jp
Kyoto University
Kyoto, Japan

Keishi Tajima
tajima@i.kyoto-u.ac.jp
Kyoto University
Kyoto, Japan

ABSTRACT

In this paper, we propose a ranking method for keyword-based book search systems. A user issues a query consisting of keywords describing the contents of the book, and the system returns a ranked list of candidate books. Because we do not have full text data of all the books, we use a database of brief descriptions of books in the market currently or in the past. When such brief descriptions are only available, some query keywords may not appear in the description of the book the user is looking for. To solve that problem, our method ranks books in two steps. We first generate relaxed queries by removing some keywords from the given original query, and rank them based on how likely the remaining keywords appear in the brief descriptions. We then retrieve matching books for each query, find words in the description that are the most similar to the removed keywords, and rank the books based on that similarity. By combining these two rankings, i.e., the ranking of relaxed queries, and the ranking of books matching with each query, we produce the final ranking. In this paper, we focus on the ranking method for the second step. Our experiment shows that our method is effective when the original query includes many keywords that do not appear in the description of the target book.

1 INTRODUCTION

We sometimes want to search for a book based on a vague memory on the contents of the book. To help such users, most libraries have a service called "reference service", where an expert librarian formulate queries consisting of keywords describing the contents, and issue them to some book databases. We usually do not have full text data of all the books, and only brief descriptions of the books are stored in the database. Such a brief description of the target book may not include all the keywords given by the user. In addition, the description by the user based on their vague memory may include mistaken words, which do not appear in the description in the database, of course. As a result, the librarian has to manually formulate many queries and try them one by one.

National Diet Library of Japan archives such questions and answers collected from libraries in Japan into Collaborative Reference Database. Many questions in it are very vague descriptions of the stories in books the users read long before, e.g., in their childhood. Similar type of questions are also found in many QA sites.

In this paper, we propose a method to support retrieval of books in such situations. Because given queries in such situations often include many words that do not appear in the descriptions of the target books in the database, we need to use relaxed queries produced by removing some words from the original query. In our method, we generate such queries, and rank the books matching with these queries in two steps. We first generate all relaxed queries by using every subset of the given set of query keywords, and rank them based on how likely the remaining keywords appear in the brief description of the target book. For each query, we then retrieve matching books, and for each book, we find words in their description that are the most similar to the removed words, and rank the books based on that similarity. By combining these two rankings, i.e., the ranking of relaxed queries, and the ranking of books matching with each query, we produce the final ranking.

These two steps roughly corresponds to the two main types of query keywords missing in the database description. The most frequent type is keywords that do not happen to appear in the database description simply because the short database description only includes very important keywords. In the first step, we give higher ranks to queries without keywords that are less likely to appear in the short database descriptions. We have proposed a ranking method for this part in our previous paper [6].

The other frequent type is keywords that are mistaken by the user. In the second step, we give higher ranks to books that include words that are likely to be mistaken for the removed keywords. Notice that simple similarity is not appropriate for finding such words. For example, even if a word is very similar to the removed keyword, if that word is very well-known and more famous than the removed keyword, the user is less likely to mistake it for the removed keyword which is less famous. We, therefore, give it a low score. In this paper, we propose a ranking method for this part.

2 RELATED WORK

Many Web search engines support the function of query suggestion. Many existing methods for query suggestion use the information on the choices made by users in the past that are recorded in query logs [1–3, 7, 9]. However, we focus on queries including mistaken words because of the vague memories, and queries including the exactly same errors rarely exist in the query log.

Elsweiler et al. [4] reported what kind of attributes of emails people remember and use for refinding the emails. Teevan also reported how people recall and reuse results of Web search queries [8]. Their experiments targeted emails and results from Web search engines, while we focus on the book search. In addition, they did not discuss the problem of mistaken query keywords.

Kim et al. [5] 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, we focus on mistaken words in the book search.

3 PROPOSED METHOD

In this section, we first briefly explain the method for the first step proposed in [6], then explain the method for the second part.

The outline of the method for the first part is as follows. First, each word in the query 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 estimate it based on statistics we obtained from the archive of a reference service explained before or 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 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.

We next explain our method for the second step. Our method is based on the following two assumptions:

- Very well-known words are less likely to be mistaken.
- People are not likely to mistake some well-known word for a less-known word, and not likely to mistake some less-known word for a well-known word.

Based on these assumptions, we define $m(a, b)$, which represents how likely people are to mistake a for b, as follows:

$$m(a, b) = |H(a) - H(b)|^{-1} \cdot |H(a)|^{-1}$$

where $H(x)$ denotes the number of books in the database whose description includes the word x. We use it to approximate the well-known degree of the word x. Our $m(a, b)$ is inversely proportional to the well-known degree of a and also inversely proportional to the difference between well-known degree of a and b.

We next define the score given to a book description D when the set of removed keywords is W, denoted by $M(D, W)$ as follows:

$$M(D, W) = \frac{1}{|W|} \sum_{w \in W} \max_{d \in D} \log_2 m(d, w).$$

For each word w in W, we calculate $m(a, w)$ with the most similar word d, and take their average over all w. We rank the books matching with a query generated by removing W by this score.

4 EXPERIMENTS

The second step of our method, i.e., the ranking of books within the result of each relaxed query, is not very important when the number of books matching with the relaxed query is small. It is important when the relaxed query that matches with the target book also matches with a large number of books. It typically happens when the number of keywords in the query is small. We show the results of the second part for several cases where the second part was important. The results of the first step for the overall experiment is shown in [6].

Below are the queries (originally in Japanese) in the four cases where the second part was important, and the ranking before and after applying our method in these cases.

- Original: thief, prison, treatment, good, caught, go to
  Relaxed query matching with the target book: thief
  Ranking: 680 → 462
- Original: siblings, treasure, show, explain
  Relaxed query matching with the target book: treasure
  Ranking: 211 → 207
- Original: mother bear, kid bear, seal, polar bear, friend
  Relaxed query: seal, polar bear
  Ranking: 2 → 3
- Original: teacher, carrot, school lunch, curry, hate
  Relaxed query: teacher, carrot, school lunch
  Ranking: 2 → 2

In the first two cases, the original ranking was very low. In these cases, our method could improve the ranking of the target book. In the last two cases, the original ranking was very high. In these cases, our method did not deteriorate the ranking.

5 CONCLUSION

In this paper, we proposed a method of ranking book descriptions for a relaxed query where some keywords are removed. We find words in the descriptions which are likely to be mistaken for the removed keyword, and rank the descriptions based on their likeliness of mistaking. This ranking is important in our system when the relaxed query matches with many books, and in such cases, our method can improve the ranking of the target book.

REFERENCES

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

[2] 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*, pages 875–883, 2008.

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

[4] D. Elsweiler, M. Baillie, and I. Ruthven. Exploring memory in email refinding. *ACM Trans. Inf. Syst.*, 26(4):21:1–21:36, Oct. 2008.

[5] Y. Kim, J. Seo, W. B. Croft, and D. A. Smith. Automatic suggestion of phrasal-concept queries for literature search. *Information Processing & Management*, 50(4):568–583, 2014.

[6] M. Kyozuka and K. Tajima. Ranking methods for query relaxation in book search. In *Proc. of IEEE/WIC/ACM International Conference on Web Intelligence*, pages 466–473, Dec. 2018.

[7] 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*, pages 971–980, 2015.

[8] J. Teevan. How people recall, recognize, and reuse search results. *ACM Trans. Inf. Syst.*, 26(4):19:1–19:27, Oct. 2008.

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