JCSE Journal of Computing Science and Engineering

Regular Paper

Journal of Computing Science and Engineering,
Vol. 6, No. 2, June 2012, pp. 161-167

Automatic Suggestion for PubMed Query Reformulation

Luu Anh Tuan* **and Jung-jae Kim**
School of Computer Engineering, Nanyang Technological University, Singapore
luuat@ntu.edu.sg, jungjae.kim@ntu.edu.sg

Abstract
Query reformulation is an interactive process of revising user queries according to the query results. To assist biomedical researchers in this process, we present novel methods for automatically generating query reformulation suggestions. While previous work on query reformulation focused on addition of words to user queries, our method can deal with three types of query reformulation (i.e., addition, removal and replacement). The accuracy of the method for the addition type is ten times better than PubMed's "Also try", while the execution time is short enough for practical use.

Category: Smart and Intelligent Computing

Keywords: PubMed; Query reformulation; Statistical approach; Query suggestion

I. INTRODUCTION

With a huge and rapidly growing literature, it is crucial for biomedical researchers to efficiently locate relevant published knowledge. However, they often find it difficult to identify appropriate queries for search engines like PubMed. For example, PubMed might return too many results for a generic keyword, including many irrelevant articles, and return too few results for a specific query, missing many relevant articles. Researchers continue to revise their queries until they get what they want. In fact, 47% of all PubMed queries are followed by a new subsequent query [1]. In this paper, we present a novel approach to query reformulation, which suggests 'likely' new queries for a given user query. PubMed is the public search engine for the biomedical literature database MEDLINE, consisting of more than 19 million citations, and is accessed by millions of users each day. It helps users in query reformulation, by providing a functionality called "Also try" (its name recently changed to "Related searches"), which suggests five new queries that are more specific than the given user query. These specific queries result from the addition of related words to the user query, where the added words are selected from the query logs of the search engine. However, we found that this "Also try" functionality performs poorly against an available set of PubMed query logs (see Section III for details). Moreover, it deals only with addition of words, but not with removal and replacement of words from user query strings. Our methods address all the three types of query reformulation; and the performance for the addition type is ten times better than that of PubMed's "Also try" function.

II. RELATED WORKS

Finding candidate terms for interactive query reformulation can be approached in many ways. One approach is to analyze search engine logs [2, 3]. In search engine logs, each query may be recorded along with the uniform resource locator (URL), which the user selected among the query results. Beeferman and Berger [4] built a bipar-

Open Access http://dx.doi.org/10.5626/JCSE.2012.6.2.161 http://jcse.kiise.org

Received 24 February 2012, **Accepted** 18 May 2012
*Corresponding Author

 pISSN: 1976-4677 eISSN: 2093-8020

tite graph for the pairs of query and URL, and applied a clustering algorithm to the graph to group related queries into a cluster, where the other queries in the cluster of the user query can be suggested to the user.

One advantage of this method is its low CPU computational cost, because it does not use actual content of Web pages and search documents, but only the co-occurrences between the queries and URLs. However, the method does not consider the fact that content words can provide useful information for query suggestion which can contribute to the precision of query reformulation.

Lu et al. [5] proposed a method to automatically produce query suggestions, based on query log analysis. It identifies the most popular queries that contain the initial user search term, and suggests them for query reformulation. This work is the basis for the "Also try" functionality of PubMed. Our approach differs from their method, in that it analyzes the text of search results, to select the most related terms. Another approach to query reformulation is to analyze the content of Web pages. Johnson et al. [6] extracted bigrams and trigrams from Web pages to find suggestion candidates. Kraft and Zien [7] used the "anchor text" of hyperlinks for the identification of potential query expansion terms. These two methods are designed for query expansion, that is, addition of words to user queries, but cannot be used for word removal and replacement. Their methods are computationally expensive, as they retrieve and analyze the full content of documents, while long response times of search engines is now not acceptable.

We may also use ontologies for query reformulation [8], although they perform best for relatively small, static and well-defined domains, not for large and generic domains [9].

Pseudo-relevance feedback (or blind feedback) is a technique for improving retrieval accuracy [10-12]. The basic idea of pseudo-relevance feedback is to assume that a small number of top-ranked documents in the initial retrieval results are relevant, and to select from these documents related terms, and add them to the query to improve query representation through query expansion, which generally leads to improvement of retrieval performance. Our approach to the addition of words to the original query is related to this idea, in that it also selects related terms from the top-ranked documents. However, our approach goes further, to compare the documents with a query-independent corpus, to filter query-specific terms.

III. DATA ANALYSIS

In this study, we obtained a single day's query logs from PubMed (ftp://ftp.ncbi.nlm.nih.gov/pub/wilbur/DAYSLOG/), where the day is described as "a typical day" [13], but the information of date is not provided, for confidentiality reasons. The log file contains 2,996,302 queries, each of which includes user ID, timestamp and query string. From the file, we extracted 2,304,507 pairs of queries from the same user ID within a one hour timeframe, where a pair consists of an initial query and a revised query. If the user reformulates a query multiple times to obtain the final query, we separate it into a set of pairs. We consider the following three types of query reformulation:

- Addition: One or more words are added to the initial query.
- Removal: One or more words are removed from the initial query.
- Replacement: One or more words in the initial query are replaced with new words.

Table 1. Percentage of reformulation categories

Category	Percentage (%)
Replacement	12.84
Addition	10.17
Removal	5.23
Others	71.76

Table 1 shows the ratio of the three types in the dataset, where the three types altogether amount to about 28%. Furthermore, we found that in more than 60% of pairs of the three types, the user reformulated (i.e., added, removed, or replaced) only one word. In this study, therefore, we focus on the three types of query reformulation for a single word, which will be presented in Section IV. The 'others' category includes query pairs, for example, whose reformulation requires a combination of the three basic reformulations (48%), indicates complete change of the initial query (24%), and involves permutation of query terms (22%). Our study of the three basic reformulations will be useful to deal with the combination of basic reformulations, while further study is required to deal with the complete changes and permutations. Please see Section VI for details.

A. Evaluation of PubMed "Also Try"

We conducted an analysis to estimate the accuracy of the "Also try" functionality of PubMed. Because this function only considers the addition type of query reformulation, we used the cases of the "addition" category in the log file, taking 1,000 randomly selected pairs into consideration. For a given pair of initial and revised query strings, we submitted the initial query string to PubMed to find its five suggestions for the query, and then checked if the revised query is found among the five suggestions. As a result, the revised query is found among the five suggestions in only 0.26% of the cases.

IV. METHODS

We implemented three modules for three corresponding reformulation types, and on top of them developed a classification module, called Reformulation classification, which predicts by which type a given query will be reformulated, as depicted in Fig. 1. The classification module classifies each initial query into one of the following four classes: "addition", "removal", "replacement" and "others". If a query is classified into "others", our reformulation system will ignore it; otherwise, it will be passed to the corresponding module for handling.

A. Reformulation Classification

We used three machine learning methods for the Reformulation classification: Naive Bayes [14], maximum entropy classifier [15] and support vector machine [16], which are based on the following features: 1) the length of query string (i.e., the number of words); 2) the average length of words in characters; 3) salient words in the query string; and 4) the hit rate (i.e., the number of search results) returned by PubMed. These features are based on our observations, such as the following examples:

- If the initial query has only one word, the user usually adds an additional word.
- If the length of the initial query is greater than 8, the user usually removes words from the query.
- If the query contains at least one noun whose length is less than 4 characters such as "bp2" and "HIV", the user usually adds one more word, or replaces this word by another one.
- If the hit rate is less than 20, the user usually removes words from the initial query. Conversely, if the hit rate is greater than 100,000, the user often adds more words.

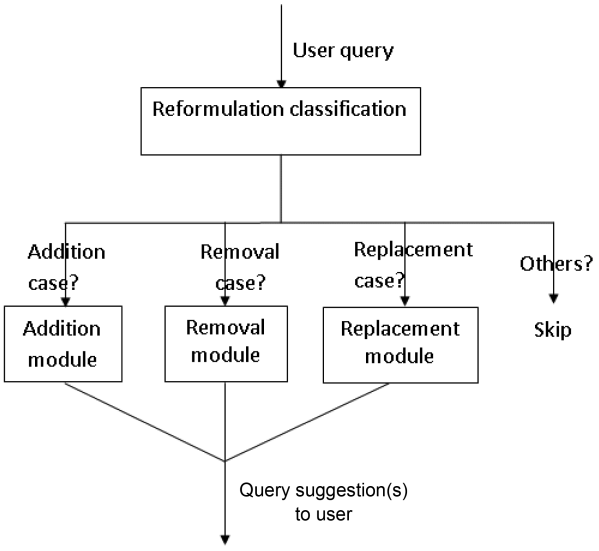

Fig. 1. Workflow of query suggestion system.

Note that the proposed features are not completely independent from each other, which violates the conditional independence assumptions for the Naive Bayes (NB) classifiers. We used an NB classifier for comparison.

B. Addition

Users generally add more words to the initial search query when the search results are too many and too general, in order to narrow down the search space. Our idea for resolving the addition type is: words that appear more frequently in the search results than in the whole literature should be more likely to be added to the initial search queries. We present a novel statistical method that computes the relative frequency of words. Since it is impractical to use the whole literature, we obtained a representative subset of the literature, by using a generic search term "gene", which is one of the most frequent words in PubMed query logs [1], to retrieve the titles and abstracts of the first 2,000 results. We call the set of the resultant documents the generic corpus (GC). As for the search results of the initial user query, since some queries return a great number of results, we collect the titles and abstracts of only the first 100 results of the user query, called the domain corpus (DC). We can then calculate the relative frequency of the word α (namely, f_α) as follows:

$$f_\alpha = \frac{\sigma^\alpha_{DC} / N_{DC}}{\sigma^\alpha_{GC} / N_{GC}},$$

where σ^α_Σ indicates the number of occurrences of the word α in the corpus Σ, and N_Σ indicates the total number of occurrences in the corpus Σ. f_α is high when the frequency of α in the DC is high, whereas its frequency in the GC is low. In other words, the words with high f values frequently co-occur with the initial query, thus having close relationships with the initial query. The addition of such closely related words would make the initial query more specific and focused. Following the convention of the "Also try" function, our method adds the five words with the highest f values to the initial query, to propose five suggestions.

C. Removal

The users would remove some words from the initial query when they receive little or no result, so that the new query becomes less specific and thus returns more results. We consider the following three heuristics for predicting the word to be removed:

- Last word: To remove the last word of the initial query. This method is based on the observation that the user usually removes the last word of an initial

Table 2. Accuracy of classification methods

Method	Accuracy (%)
Naive Bayes	62.81
Maximum entropy	64.13
Support vector machine	65.44

query, when they do the query reformulation.

- Word with lowest hit rate: To remove the word with the least hits, which might be the most specific word in the query string. We submit each word in the initial query to PubMed to find the word with the smallest number of results.
- Remaining query with highest hit rate: To remove the word whose removal returns the most hits. We submit the query without each word to PubMed, and find the one with the largest number of results.

Note that the removal module suggests at most three new queries, which are the results of the three heuristics above.

D. Replacement

When the search results are too few, the users can remove some specific words from the queries, but they also can replace the specific words with other words, which are less specific and more relevant to the rest of the query strings. Our approach to the replacement cases is to combine our methods for the addition and removal cases; in other words, we find the word to be replaced using our methods for the removal cases, and find the new word to be added using our method for the addition cases. As a result, our replacement module suggests five new queries for an initial user query.

V. Experimental Results

We compared three methods for the Reformulation classification: NB, maximum entropy and support vector machine, whose accuracy is shown in Table 2. In each method, we used 10-fold cross-validation. Since support vector machine shows the best accuracy, we used it for the classification module.

Three proposed methods for the removal cases described in Section IV-C are also compared in terms of accuracy, which we measure as follows: if a suggestion by the methods is identical to the revised query for the initial query, it is correct. We estimated the accuracy of the removal methods by using the removal cases in our dataset, and the evaluation results are shown in Table 3. The Word with lowest hit rate method shows the best performance, which supports the hypothesis that users remove the most specific words to get more results in removal cases.

Table 3. Accuracy of three heuristics for removal

Method	Accuracy (%)
Last word	59.47
Word with lowest hit rate	60.11
Remaining query with highest hits	42.92

Table 4. Accuracy of three heuristics used to predict the position of replaced word

Method	Accuracy (%)
Last word	52.31
Word with lowest hit rate	60.98
Remaining query with highest hits	35.05

We also compared three methods for predicting the word to be replaced, as described in Section IV-D (these methods are, in fact, the three methods used for removal cases). In this comparison, we used the replacement cases in the log file for the experiment, and the results are shown in Table 4. The Word with lowest hit rate again has the best accuracy, and thus we use this with our addition method described in Section IV-B for the replacement module.

In the cases of addition and replacement, their methods suggest five new queries for an initial query. Thus their accuracy is measured as follows: If the five suggestions include the revised query for the initial query, the module works correctly. Table 5 shows the accuracy and average execution time of the three modules and the whole system. The performance of the overall system, which includes the reformulation classification module, is affected by the accuracy of the classification module, as well as by the accuracy of the three modules. The average execution time of the whole system is less than one second, when we tested our system on a computer with 2.67 GHz CPU and 3 GB memory. It might show better speed if we deploy the system in a high-performance computing environment.

The accuracy of the addition module is more than ten times better than that of the "Also try" functionality of PubMed (0.26%). This result shows that content analysis is more effective than query log analysis, and that our method is fast enough for practical use.

VI. DISCUSSION

Our goal in this study is to aid users when they like to revise their queries. We do not claim that the revised query is always more appropriate than the original query, allowing 'trial-and-error' by users. If the revised query is not better than the original query, the user might like to try another query, which is out of the scope of our paper.

Table 5. A summary of evaluation results

Module	Accuracy (%)	Average execution time (sec)	Minimum time (sec)	Maximum time (sec)	Standard deviation of execution time (sec)
Classification	65.44	0.03	1×10^{-4}	0.12	0.014
Addition	3.08	1.03	0.29	1.68	0.310
Removal	80.39	0.12	0.02	0.27	0.038
Replacement	3.97	1.29	0.36	2.04	0.472
Whole system	2.38	0.86	0.02	2.08	0.693

Table 6. Subcategorization of 'others' category

Category	Percentage (%)
Permutation + replacement	8.47
Permutation + addition	9.12
Permutation + removal	4.46
Addition + replacement	16.09
Removal + addition	14.68
Addition + replacement	16.79
Complete change	23.90
Other changes	6.49

Table 7. A summary of evaluation results for a revised system tolerating permutation

Module	Accuracy (%)
Classification	65.44
Addition	3.51
Removal	81.70
Replacement	4.33
Whole system	2.52

As pointed out by Islamaj-Dogan et al. [1], there are a lower abandonment rate and a higher reformulation rate for PubMed, implying that PubMed users are more persistent in pursuing their information needs, than users of other search systems [17, 18]. This issue would possibly be addressed by considering the sequence of query reformulations by a user (see below for further discussion). In our query reformulation system, we assume that a query consists of terms that are implicitly connected with AND operators, not taking into account the Boolean operators (e.g., AND, OR, NOT), due to the observation that only 0.32% of the queries in our dataset include Boolean operators. Even though we revise the evaluation criteria to consider Boolean operators, the overall performance hardly increases (from 2.38% to 2.40%). For example, in the addition case, if the initial query is δ and the revised query is "δ *AND* w", and if w is the word predicted by our system, then we may count our suggestion "δ w" as correct. But, we did not consider this effect in the experimental results reported in the previous section because it has almost no effect. To further characterize the query pairs of the others category, we analyzed if their changes match any combination of the three basic reformulation categories. Table 6 shows the analysis results. 'Permutation' means the two queries have the same set of words, where the word orders are different. The subcategory 'complete change' in the table, which is the most popular (23.9%) but not dominant, means that the revised query is completely different from the original query. To deal with this case, we might need to understand users' behaviors [19, 20] and goals [21] in information retrieval, which are out of the scope of our research. Furthermore, around 22% query pairs in the 'others' category are reformulated with the combination of 'permutation' and one of the three basic reformulations. Since the terms in a query without Boolean operators are implicitly connected with AND operators, the word order does not affect the search result. Therefore, we revised the evaluation criteria to tolerate permutation in the revised query, which means that if the word set of a suggestion (e.g., "lung cancer human") is identical to that of the user's revised query (e.g., "human lung cancer"), we count the suggestion as correct, regardless of word order.

Table 7 shows the evaluation results according to the new criteria. In addition, more than 47% of the 'others' category is reformulated with the combination of two of the three basic reformulations (i.e., addition + replacement). We may address these cases by combining our methods for the basic reformulations. We leave this issue for future work. A user may revise a query more than one time consecutively, as PubMed users are more persistent in pursuing their information needs than users of other search systems [17, 18]. We thus need to consider the sequence of reformulations. Table 8 shows the statistics of two consecutive reformulations, where the sequences of three basic reformulations amount to 25.5%. Based on the statistics, we might understand that, after a user adds a term to the initial query, there is a higher probability of adding another term, than of removing a term from the revised query.

Also, after a user removes a term from the initial query,

Table 8. Sequence of two reformulations

Sequence	Percentage (%)
Addition → addition	3.11
Addition → removal	1.30
Addition → replacement	4.09
Removal → removal	1.98
Removal → addition	2.67
Removal → replacement	2.55
Replacement → replacement	3.25
Replacement → addition	3.39
Replacement → removal	3.16
Others	74.50

there is slightly less probability of removing another term than the addition or replacement reformulations. As a future work, we plan to extend this analysis to include more classes and consider semantic categories of query terms.

VII. CONCLUSION

We proposed a novel approach to automatic generation of query reformulation suggestions for PubMed. While previous work on query reformulation focused only on the addition type, our method covers three reformulation categories: addition, removal and replacement. In particular, its performance on the addition type is ten times better than that of the "Also try" function, while the execution time is short enough for practical use. As a future work, we will develop a method for reformulation sequences and semantic categorization. We will also study the usage of user behaviors for automatic query reformulation.

REFERENCES

1. R. Islamaj-Dogan, G. C. Murray, A. Neveol, and Z. Lu, "Understanding PubMed user search behavior through log analysis," *Database: the Journal of Biological Databases and Curation*, vol. 2009, article id. bap018, 2009. doi: 10.1093/database/bap018.
2. A. Spink, B. J. Jansen, and H. C. Ozmultu, "Use of query reformulation and relevance feedback by excite users," *Internet Research*, vol. 10, no. 4, pp. 317-328, 2000.
3. P. Wang, M. W. Berry, and Y. Yang, "Mining longitudinal web queries: trends and patterns," *Journal of the American Society for Information Science and Technology*, vol. 54, no. 8, pp. 743-758, 2003.
4. D. Beeferman and A. Berger, "Agglomerative clustering of a search engine query log," *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Boston, MA, 2000, pp. 407-416.
5. Z. Lu, W. J. Wilbur, J. R. McEntyre, A. Iskhakov, L. Szilagyi, "Finding query suggestions for PubMed," *American Medical Informatics Association (AMIA) Annual Symposium Proceedings*, 2009, pp. 396-400.
6. D. Johnson, V. Malhotra, and P. Vamplew, "More effective web search using bigrams and trigrams," *Webology*, vol. 3, no. 4, article 35, 2006.
7. R. Kraft and J. Zien, "Mining anchor text for query refinement," *Proceedings of the 13th International Conference on World Wide Web*, New York, NY, 2004, pp. 666-674.
8. J. Akahani, K. Hiramatsu, and T. Satoh, "Approximate query reformulation based on hierarchical ontology mapping," *Proceedings of International Workshop on Semantic Web Foundations and Application Technologies*, Nara, Japan, 2003.
9. E. Meij and M. de Rijke, "Thesaurus-based feedback to support mixed search and browsing environments," *Proceedings of the 11th European Conference on Digital Libraries*, Budapest, Hungary, 2007, pp. 247-258.
10. J. J. Rocchio, "Relevance feedback in information retrieval," *The SMART Retrieval System: Experiments in Automatic Document Processing*, G. Salton, ed., Englewood Cliffs, NJ: Prentice-Hall, 1971. pp. 313-323.
11. V. Lavrenko and W. B. Croft, "Relevance based language models," *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New Orleans, LA, 2001, pp. 120-127.
12. C. Zhai and J. Lafferty, "Model-based feedback in the language modeling approach to information retrieval," *Proceedings of the 10th International Conference on Information and Knowledge Management*, Atlanta, GA, 2001, pp. 403-410.
13. J. R. Herskovic, L. Y. Tanaka, W. Hersh, E. V. Bernstam, A day in the life of PubMed: analysis of a typical day's query log," *Journal of the American Medical Informatics Association*, vol. 14, no. 2, pp. 212-220, 2007.
14. I. Rish, "An empirical study of the Naive Bayes classifier," *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, Seattle, WA, 2001, pp. 41-46.
15. A. McCallum, D. Freitag, and F. C. N. Pereira, "Maximum entropy Markov models for information extraction and segmentation," *Proceedings of the 17th International Conference on Machine Learning*, Stanford, CA, 2000, pp. 591-598.
16. C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273-297, 1995.
17. B. J. Jansen, D. L. Booth, and A. Spink, "Determining the informational, navigational, and transactional intent of Web queries," *Information Processing and Management*, vol. 44, no. 3, pp. 1251-1266, 2008.
18. S. Y. Rieh and H. Xie, "Analysis of multiple query reformulations on the web: the interactive information retrieval context," *Information Processing and Management*, vol. 42, no. 3, pp. 751-768, 2006.
19. C. Liu, J. Gwizdka, J. Liu, T. Xu, and N. J. Belkin, "Analysis and evaluation of query reformulations in different task types," *Proceedings of the American Society for Information Science and Technology*, Pittsburgh, PA, 2010.
20. M. C. Burton and J. B. Walther, "The value of web log data

in use-based design and testing," *Journal of Computer-Mediated Communication*, vol. 6, no. 3, 2001.
21. N. Kirtsis and S. Stamou, "Query reformulation for task-oriented web searches," *Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, Lyon, France, 2011, pp. 289-292.

Luu Anh Tuan

Mr. Anh Tuan LUU received his M.Sc. degree in Computer Science from the National University of Singapore, on Security Model checking. Since 2011, he has been a research associate at the Centre for Advanced Information Systems at Nanyang Technological University, Singapore. His research interests include formal methods, security protocol verification, semantic web and bioinformation retrieval.

Jung-jae Kim

Jung-jae Kim is an assistant professor at the School of Computer Engineering, Nanyang Technological University, Singapore. He received his B.S., M.S., and Ph.D. degrees in Computer Science from KAIST, Korea, in 1998, 2000, and 2006, respectively. His research areas include biomedical text mining and ontology engineering.