Retrieval of Biomedical Documents by Prioritizing Key Phrases

Kevin Hsin-Yih Lin, Wen-Juan Hou and Hsin-Hsi Chen

Department of Computer Science and Information Engineering,
National Taiwan University
Taipei, Taiwan, 106

E-mail: {hylin, wjhou }@nlg.csie.ntu.edu.tw; hhchen@csie.ntu.edu.tw

Abstract

In this paper, we present an approach for retrieving relevant articles from the biomedical corpus. Our first run considered four kinds of operators as query expansion. The operators are phrase, mandatory, optional and synonym set. The second run lowered the ranking of documents which contained query terms only in their MeSH fields. The results of the official runs and further experiments were listed.

1. Introduction

The Ad Hoc Retrieval Task of TREC 2005 Genomics Track uses topics which are based on five generic topic templates. This is different from the retrieval task of 2004, which uses freeform topics. The five templates, as mentioned in the official task description, are (1) Find articles describing standard methods or protocols for doing some sort of experiment or procedure; (2) Find articles describing the role of a gene involved in a given disease; (3) Find articles describing the role of a gene in a specific biological process; (4) Find articles describing interactions (e.g., promote, suppress, inhibit, etc.) between two or more genes in the function of an organ or in a disease; (5) Find articles describing one or more mutations of a given gene and its biological impact [5].

For the retrieval task, we implemented an information retrieval system which supports phrase searching and BM25 scoring. Our system first extracted key terms from topic narratives by pattern matching. The Entrez Gene database and MeSH database were used for query expansion. We did not rank documents directly by BM25 score. Instead, we devised a method to identify more important phrases in a query and gave these terms higher priority over the less important phrases, regardless of their BM25 scores. We indexed only the TI, AB, MH, RN and GS fields of the Medline entries. The stop word list of PubMed was used to filter out stop words. We also used the Porter stemmer to stem all words.

2. System Description

Our retrieval system is an extension of the BM25 document scoring scheme. The BM25 formula is defined as

$$\mathrm{BM25}(q,d) = \sum_{t \in Q} q_t \times \ln\left(\frac{D - df_t + 0.5}{df_t + 0.5}\right) \times \frac{freq_{t,d} \times (1 + k_1)}{freq_{t,d} + k_1\left((1-b) + b \times \frac{dl_d}{avdl}\right)}$$

where BM25*(q,d)* is the BM25 score of document *d* for the query *q*, q_t is the frequency of the term *t* in *q*, *D* is the total number of documents, df_t is the document frequency of *t*, $freq_{t,d}$ is the term frequency of *t* in *d*, k_1 is a parameter, *b* is a parameter, dl_d is the document length of *d* and *avdl* is the average document length of all the documents [4]. We used the same k_1 and b values as those used by Büttcher *et al*. in TREC 2004 Genomic Tracks [3]. The parameters were set to be $k_1 = 1.2$ and $b = 0.75$.

In our retrieval system, we added phrase, mandatory, optional and synonym set operators. We evaluated an expanded query by creating different permutations of a query out of the query term synonyms, instead of adding the synonyms to the original query. To evaluate a query, our system went through three iterations, each time relaxing some constraints of the query.

2.1 Operators

Our query used four kinds of operators: phrase, mandatory, optional, and synonym set. The operators are represented by the brackets ( ), < >, [ ], and { } respectively. The optional, mandatory and phrase operators contained one or more query terms. The synonym set operator contained one or more of the three other operators. For instance, a sample query is "{(ribosomal protein l11) (cd220)} {<cancer> <tumor>}". In the query, "(ribosomal protein l11) <cancer>" is the original query, which looks for documents containing the phrase "ribosomal protein l11" and the term "cancer". "cd200" and "tumor" are the synonyms of "ribosomal protein l11" and "cancer" respectively.

In order for a document to satisfy a phrase operator, the exact phrase contained in the phrase operator must appear in the document. If a document contained such a phrase, the weight of the phrase was computed by summing up the BM25 score of each term in the phrase. If a query contained a phrase operator, then a document which did not satisfy the phrase operator was considered as irrelevant regardless of what the rest of query was.

The mandatory operator behaved very much like the phrase operator, except that terms inside a mandatory operator did not have to occur next to each other or in any particular order in a document. A document satisfied a mandatory operator if the document contained all the terms inside the operator. The weight of a mandatory operator was the sum of the BM25 scores of all the terms inside the operator. Again, if a query contained a mandatory operator, then a document which did not satisfy the mandatory operator was seen as irrelevant regardless of what the rest of query was.

The optional operator was the least restrictive operator. For each document, the weight of an optional operator was the sum of the BM25 scores of all the terms that were both inside the operator and appeared in the document.

The final score of a document was the sum of BM25 scores of all the operators.

2.2 Query Expansion

The synonym set operator was used to expand a query. We did not put all synonyms into a single query. Instead, we constructed different permutations of a query, each containing a different member of the synonym set. If a query contained more than one synonym set, then the number of queries generated was the product of the sizes of the synonym sets. The relevance score of a document was its maximum relevance score over all the query permutations.

Continuing with the previous query example, we would obtain the following four queries after query expansion:

- (ribosomal protein l11) <cancer>
- (ribosomal protein l11) <tumor>
- (cd220) <cancer>
- (cd220) <tumor>

2.3 Three Iterations

Since the phrase and mandatory operators were highly restrictive, the number of documents retrieved could be very small when a query contained too many phrase and mandatory operators. To increase the number of documents retrieved, we evaluated a query in three iterations, each time reducing some constraints of the query. The first iteration used the original set of expanded query permutations. In the second iteration, we converted phrase operators to mandatory operators. In the last iteration, we converted all operators to optional operators.

As an example, the query permutations of the query "{(ribosomal protein l11) (cd220)} {<cancer> <tumor>}" at each of the three iterations are shown below:

Iteration 1:
- (ribosomal protein l11) <cancer>
- (ribosomal protein l11) <tumor>
- (cd220) <cancer>
- (cd220) <tumor>

Iteration 2:
- <ribosomal protein l11> <cancer>
- <ribosomal protein l11> <tumor>
- <cd220> <cancer>
- <cd220> <tumor>

Iteration 3:
- [ribosomal protein l11] [cancer]
- [ribosomal protein l11] [tumor]
- [cd220] [cancer]
- [cd220] [tumor]

During query evaluation, the documents proposed (i.e., documents with BM25 score greater than 0) in a preceding iteration always have higher ranks than the documents proposed by a succeeding iteration, regardless of the relevance weight of the documents. In the first and second iterations, the documents proposed by the original query (e.g. the query containing "ribosomal protein l11" and "cancer" in our persisting example) always have higher ranks than the documents proposed by other permutations of the original query. In summary, the proposed documents can be grouped into the following five categories in descending order of priority:

1. Documents proposed by the original query in Iteration 1
2. Documents proposed by the other query permutations in Iteration 1
3. Documents proposed by the original query in Iteration 2
4. Documents proposed by the other query permutations in Iteration 2
5. Documents proposed by all query permutations in Iteration 3

Within each category, documents are ranked by the relevance weight. If a document belongs to more than one category, then the category with the higher priority is used.

3. Query Formulation

To formulate a query that can be accepted by the retrieval model in Section 2, we took the topic narratives and went through the steps of template type identification, gene name synonym expansion, query phrase identification and query phrase synonym expansion.

3.1 Template Type Identification

Each topic narrative follows a specific format depending on its corresponding template. The narrative patterns of the five template types are listed below:

1. Describe the procedure or methods for *experiment description*.
2. Provide information about the role of the gene *gene name* in the disease *disease name*.
3. Provide information on the role of the gene *gene name* in the process of *biological process*.
4. Provide information about the genes *gene names* in *function of organ or disease*.
5. Provide information about *gene mutation* and its/their *biological impact*.

In an actual narrative, the strings in italic, which we will refer to as slots, are substituted by terms which the topic is about. For example, one of the narratives belonging to Template 2 is “Provide information about the role of the gene IDE gene in the disease Alzheimer's Disease”. The *gene name* slot is filled by “IDE gene” and the *disease name* slot is filled by “Alzheimer’s Disease”. We used the above five patterns to extract the terms which appear in the location of the slots from the narratives.

We did not use the strings which appear in the slots to formulate queries directly. Instead, we performed further processing to identify individual gene names and query phrases.

3.2 Gene Name Recognition

Gene names appear primarily in the *gene name*, *gene names* and *gene mutation* slots of the five narrative patterns. We treated the string filling the *gene name* slot as the name of a single gene. For a string filling the *gene names* slot, we assumed that gene names are separated by commas and the word “and”. For a string filling the *gene mutation* slot, we first removed all occurrences of the word “mutation” and “mutations”. Then we used PubMed’s stopword list to remove any stopword occurring in the string. The remaining terms were considered to be a single gene name. For all three slots, any string occurring inside parentheses is seen as a synonym of the immediately preceding gene name. We also filtered out all occurrences of the word “gene”.

3.3 Gene Name Synonym Expansion

Once the gene names are identified, we used the gene_info file provided by the Entrez Gene database to find gene synonyms [2]. The Symbol, LocusTag, Synonyms, Symbol From Nomenclature Authority, and Full Name From Nomenclature Authority fields of the gene_info file were used to form synonym sets. From the gene_info file, we removed gene names which are synonyms of more than one gene names. This was done to reduce ambiguity.

For each gene name identified by the method in Section 3.2, we selected up to two synonyms. We did not use all possible synonyms, because we wanted to reduce search time. To select the best synonyms, we ranked the synonyms by the following measure:

$$Score(gene\_name, synonym) = \frac{LCS\_Length(gene\_name, synonym)}{Length(gene\_name) + Length(synonym)}$$

where *gene_name* is the original gene name, *synonym* is the synonym of *gene_name*, *Length* is a function that returns the length of a string, *LCS_Length* is a function that returns the length of the longest common subsequence of two strings. For each gene name, the two synonyms with the highest score are chosen as the synonyms for query expansion. In effect, we wanted to find synonyms that resemble the original gene name and are short.

If a gene name did not appear in the gene_info file and hence did not have readily available synonyms, we applied heuristics to create possible variants of the gene name. We converted Roman numerals from one to ten into Arabic numerals and vice versa. We also added or deleted spaces between a number and the preceding term. For example, the string “beta 2” has the variants “beta ii”, “beta2” and “betaii”.

To form the query, we used a phrase operator on each of the gene name and its synonyms. We then use the synonym set operator on the phrase operators containing the gene names.

3.4 Query Phrase Recognition and Expansion

Query phrases are the important phrases which appear in the *experiment description*, *disease name*, *biological process*, *function of organ or disease* and *biological impact* slots of the narrative patterns. A query phrase was found by finding the longest substring in the topic template which matched a phrase in the PRINT ENTRY or ENTRY fields of the MeSH descriptor file d2005.bin [1]. Once a query phrase has been found, the phrases in the PRINT ENTRY and ENTRY fields of the same MeSH term were used as the synonyms of query phrase. To form a query, we used the mandatory operators on the query phrases and their synonyms. We then use the synonym set operator on the phrase operators containing the query phrases.

4. Experiment Result

We submitted two runs for evaluation. The first run (NTUgah1) employed the methods described above. For the second run (NTUgah2), we lowered the ranking of documents which contained query terms only in their MeSH fields. The performance results are listed in Table 1.

Table 1: Overall Performance of Official Runs

Run	**P10**	**P100**	**MAP**
NTUgah1	0.3918	0.1998	0.2173
NTUgah2	0.3980	0.1996	0.2204

In Table 1, P10 is the precision after 10 documents are retrieved, P100 is the precision after 100 documents are retrieved, and MAP is the mean average precision.

To study the effects of query expansion and prioritizing documents containing key phrases, we did further experiments with different combinations of these features. Table 2 shows the results.

Table 2: Performance for Different Combinations of Features

Run ID	**Query Expansion**	**Prioritization For All Queries**	**Prioritization For Original Query**	**MAP**
Baseline	No	No	No	0.2516
Run 1	No	Yes	Yes	0.2480
Run 2	Yes	No	No	0.1876
Run 3	Yes	No	Yes	0.2180
Run 4	Yes	Yes	No	0.2032
NTUgah2 (official)	Yes	Yes	Yes	0.2204

In Table 2, Query Expansion indicates whether query expansion is used. Prioritization For All Queries means that documents containing phrases enclosed in phrase or mandatory operators in the original query or expanded queries are prioritized. Prioritization For Origin Query means that documents containing phrases enclosed in phrase or mandatory operators in the original query is prioritized. If a run uses both Prioritization For All Queries and Prioritization For Original Query, then documents that contain phrases in the original query have higher priority than documents that contain phrases only in the expanded queries. For runs without query expansion, Prioritization For Original Query and Prioritization For All Queries are the same thing, because there are no expanded queries. The baseline uses BM25 without any query expansion or prioritization.

5. Discussion

Comparing Baseline and Run 1 to Run 2, Run 3, Run 4 and NTUgah2, we observe that doing query expansion reduces the performance of the system. This can mean that our method for expanding queries is not effective or query expansion is not helpful in general for this set of corpus and queries.

We have varied results for the Prioritization For All Queries feature. Comparing Run 2 to Run 4 and Run 3 to NTUgah2, we see that Prioritization For All Queries improves the performance slightly. However, comparing Baseline to Run 1, the performance is lowered instead. One possible explanation for this is that our method for identifying important phrases in a query works better on expanded queries than the original query.

If we compare Run 2 to Run 3 and Run 4 to NTUgah2, we see that Prioritization For Original Query increases the performance. This makes sense, because the origin query should capture the need of the user best.

6. Conclusion

In this paper, we demonstrated how our system was constructed. The TI, AB, MH, RN and GS fields of the Medline entries were considered as a representation of the article. The Entrez Gene database and MeSH database were used for query expansion. Four kinds of operators, i.e., phrase, mandatory, optional and synonym set, were used in this study. Documents that contain the important phrases in the queries are ranked higher. We found out that our method of query expansion does not improve the performance, but our method of prioritizing documents does improve the result of query expansions.

References

[1] Medical Subject Headings. http://www.nlm.nih.gov/mesh/meshhome.html, 2005.

[2] NCBI Entrez Gene. http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?CMD=search&DB=gene, 2005.

[3] Stefan Büttcher, Charles L.A. Clarke, and Gordon V. Cormack. Domain-Specific Synonym Expansion and Validation for Biomedical Information Retrieval (MultiText Experiments for TREC 2004). In *Proceedings of Text REtrieval Conference 2004 (TREC 2004)*, November 2004.

[4] Stephen Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. Okapi at TREC-3. In *Proceedings of the Third Text REtrieval Conference (TREC-3)*, November 1994.

[5] TREC 2005 Genomics Track Protocol. http://ir.ohsu.edu/genomics/2005protocol.html, 2005.