Egyptian Informatics Journal 22 (2021) 75–84

Contents lists available at ScienceDirect

Egyptian Informatics Journal

journal homepage: www.sciencedirect.com

Automatically transforming full length biomedical articles into search queries for retrieving related articles

Shariq Bashir [a,*], Akmal Saeed Khattak [b], Mohammed Ali Alshara [c]

[a] The College of Arts and Sciences, Department of Computer Science, University of Nizwa, Sultanate of Oman
[b] Department of Computer Science, Quaid-i-Azam University, Islamabad, Pakistan
[c] Department of Information Technology, College of Computer and Information Sciences, Imam Mohammad Ibn Saud Islamic University, Riyadh, Saudi Arabia

ARTICLE INFO

Article history:
Received 25 October 2019
Revised 22 March 2020
Accepted 25 April 2020
Available online 16 May 2020

Keywords:
Biomedical search system
Clinical decision support system
Related citation search
Learn to rank
Information retrieval

ABSTRACT

Searching relevant articles from medical resources is an important search task in clinical decision support system. The technical contents and long length of biomedical articles make this search task more complicated than many other search tasks. Previous research on biomedical information retrieval (IR) is typically based on keyword search. In this paper we propose a new approach. Using our approach, a user can use the full article as a query. This reduces the burden on the users and generates an effective automatic query from many more useful search features. In this novel search scenario, we explore in detail several important factors for developing a successful biomedical articles retrieval system, especially focusing on how to automatically convert an article into an effective search query. Specifically, we evaluate the performance of single features with different parameter configurations, as well as combinations of these features using the techniques of learning to rank and rank fusion. Experimental results on PubMed collection show that the introduction field is the most useful feature for transforming a query. Furthermore, our experiments showed that combining multiple features can significantly improve the effectiveness of a search system.

1. Introduction

World wide web provides explosion of information related to medical and health in the form of medical resources, bibliographic databases and blogs. To diagnose a condition of a patient, physicians increasingly rely on the information available in the medical resources [4,12,20]. Unfortunately, physicians face difficulties when trying to search the information from the research articles. In most cases they end on retrieving large number of irrelevant articles. Biomedical articles have complex technical contents and structures, which can create significant difficulties for a retrieval system for searching relevant documents. Medical research articles are usually very long which create challenges for the information retrieval (IR) system for capturing the real content of a topic. These factors make biomedical articles retrieval significantly different to web search.

Currently, retrieval systems for searching biomedical articles are typical keyword based systems [31,32], such as *PubMed*. In these retrieval systems, the success of the search depends on the quality of the query words used by the physician. However, due to the long length of the article and the professional knowledge required to understand the content, selecting relevant keywords can be a difficult task [19]. In these situations, it may be better for the physician if he/she can specify his/her query as a small set of initial retrieve articles rather than using a set of keywords. In particular, having already read articles retrieved from the initial query relating to a study, the physician often interested in, *"given that these articles relating to a study, what else are relevant articles to this study*".

In this paper, we propose a new approach. Using our approach, a user can directly use the whole medial article as a query instead of just the query words. This reduces the burden on the users and generates an effective automatic query from many more useful search features. Given an article as a query, the system can utilise the full abundant information available in the article and can search many more potentially useful retrieval features. Table 1 shows an example biomedical article. Several types of information are available in this article. First, the $< TITLE >$ *(ttl)*, $< ABST >$

* Corresponding author.
E-mail addresses: shariq.bashir@unizwa.edu.om (S. Bashir), akhattak@qau.edu.pk (A.S. Khattak), mamalsharaa@imamu.edu.sa (M.A. Alshara).

Peer review under responsibility of Faculty of Computers and Information, Cairo University.

https://doi.org/10.1016/j.eij.2020.04.004
1110-8665/

(abst), *< INTR > (intr)*, *< FIGR > (figr)*, *< RSLT > (rslt)*, *< DISS > (diss)* and *< CONL > (conl)* tags indicate the title, the abstract, the introduction, the description of the figures and tables, results, discussion of results, and the conclusion, respectively. With this type of structural information, we can analyze the properties of different fields and explore the words in different fields as search features. Second, noun phrases are widely used in the medical domain to describe the concepts related to the condition. In the above example, the noun phrases are underlined. Phrases such as "*persistent symptomatic infection*", "*T-cell immunodeficiency*" and "*autosomal dominant (AD) hyper*" appear to be more promising search features than single words.

In other contribution of this paper we combine features for improving effectiveness. Among the possible feature combination techniques, a linear combination is widely used due to its simplicity and comprehensibility. Specifying the combination weight for each feature is essential for the success of this method. Typically, the weights are assigned equal values if no additional information is provided or are assigned according to human knowledge. In the case of biomedical articles search, the weights can be decided using learning techniques. An article reference section indicates its related references decided (see Table 1 for example), which could be a reasonable substitute for the real relevance judgments from the experts. In this way, it is easy to prepare a training set with thousands of queries and then optimal combination of weights can be learned with rank fusion and learning to rank.

Furthermore, in this paper, we compare many different search features extracted from the article, including words or noun phrases from different fields by applying different weights. Results of our experiments show that the words and noun phrases from the introduction field provide more effectiveness than those from the title or abstract field when used for generating queries for retrieving biomedical articles. Furthermore, after combining different features, retrieval performance can be significantly improved over single features.

2. Related work

Searching related articles from biomedical articles collection is a challenging task. This is because it is hard to choose keywords that express the user's specific relevance and an irrelevant query retrieves many bad documents. For improving the quality of search query IR researchers have studied different techniques. The techniques that the researchers studied are relevance feedback, controlled vocabulary, and background knowledge to generate better search queries [17,24].

[29] proposed a query expansion using relevance feedback for *PubMed* by using *RankSVM* to retrieve relevant documents. Their proposed system use relevance ranking by applying query expansion on PubMed. Their system first retrieves initial documents for a user's query. The user then interacts with the systems and marks and selects relevant documents while browsing retrieved results. Once the user provides relevance feedback, the system runs a relevance feedback function using *RankSVM* and re-ranks the retrieved results.

[9] used *UMLS Metathesaurus* for selecting words for query expansion. Their results showed that thesaurus based query expansion did not improve retrieval effectiveness. [1] performed query expansion experiments on *MEDLINE* abstracts by using pseudo relevance feedback methods.

[11] used ontologies and clustering for re-ranking the search results for providing relevant documents to the users. [13,11,23] used text mining techniques for computing global importance of biomedical articles using the citation information and apply it for re-ranking the initial results. The main limitation of these techniques is that these techniques cannot capture the varied hidden relevances of different users for the same query. Therefore, re-ranking the initial results using the global importance usually does not capture the personalized or users' specific information needs.

In the information retrieval area, many rank fusion techniques have been proposed to combine different document rankings. The linear combination method represents an approach that linearly combines the scores in each ranking to get the final score. [22] proposed several operators such as *CombSUM*, *CombMIN*, *CombMAX*, *CombANZ* and *CombMNZ*, which correspond to different combination strategies. Other methods included minimizing the Spearman's footrule distance to the input ranks and learning the probabilistic model from training queries [10]. Instead of focusing on the exact score or rank of a document, the *ELECTRE* method [21] considered the pairwise relations between documents.

Recently, "learning to rank" techniques have gained considerable attention for searching relevant information from medical resources [27,2,25]. In learning to rank each pair of query-documents is described with a set statistical features and then a training set is construct using previously seen queries and their relevance judgments. Then machine learning techniques are used for training a ranking model on the training set. The learned model can then be used for ranking documents for the unseen queries. IR researches have proposed several learning to rank methods, such as *Ranking SVM* [8], *RankBoost* [6], *RankNet* [3] and *AdaRank* [28]. [2] used learning to rank in biomedical literature search for retrieving relevant information for physicians for proving better care to their patients. They proposed a set of learning features and feature selection method for learning efficient models for increasing the performance of retrieving relevant biomedical literature. [25] proposed a system using learning to rank for health seekers for searching relevant information from medical resources. They used syntactic and semantic features for training learning to rank system to capture the similarity between the query retrieved documents. They evaluated their approach using 2016 CLEF eHealth dataset, and showed their approach outperformed the best method by 26.6% in *NDCG*@10.

[25] proposed two query reformulation techniques for medical professionals for retrieving relevant information from medical literature. In their first technique, they used unsupervised query expansion and pseudo relevance feedback. Their unsupervised query expansion approach only includes health related terms. In

Table 1
An example biomedical article.

<TITLE>Inborn errors of human IL-17 immunity underlie chronic mucocutaneous candidiasis</TITLE>
<ABST> Chronic mucocutaneous candidiasis (CMC) is characterised by recurrent or persistent symptomatic infection of the nails, skin and mucosae mostly by Candida albicans. CMC is common in patients with profound primary T-cell immunodeficiency, who often display multiple infectious and autoimmune diseases. Patients with syndromic CMC, including autosomal dominant (AD) hyper IgE syndrome (HIES) and autosomal recessive (AR) autoimmune polyendocrinopathy syndrome type I (APS-I), display fewer other infections. Patients with isolated CMC (CMCD) rarely display any other severe disease. We review here recent progress in the genetic dissection of these three types of inherited CMC. </ABST> <KEYWORDS> Primary immunodeficiencies, chronic mucocutaneous candidiasis, interleukin-17 immunity, Candida albicans </KEYWORDS> <INTR>. . . . .</INTR> <RSLT>. . . . .</RSLT> <FIGR>. . . . .</FIGR> <DISS>. . . . .</DISS> <CONL>. . . . . .</CONL> <REFR>. . . . . .</REFR>

their second technique, they proposed supervised approach for query expansion using learning to rank. Their second approach uses deep neural network for extracting relevant candidate terms using a weighted relevance ratio by measuring the importance of each term in relevant documents. Their experiments show that removing non–health related terms improves the effectiveness of search system.

Table 2 provides the summary of ranking techniques used in the related work. As we can see from the comparison summary, all related approaches depend on the user for providing initial keywords of a query. Our work is different to related approaches. In this article, we consider a novel scenario, where the user directly poses the full article as the query instead of the query. This reduces the burden on the users and generates an effective automatic query from many more useful search features. Given an article as a query, the system can utilise the full abundant information available in the article and can search many more potentially useful retrieval features. The idea of automatically transforming a full article into search query is motivated by the availability of many useful words in the document. These words contain strong relationship with the topic of article, which makes it possible to retrieve related documents. However, due to the length of the research article and technical content selecting suitable words manually can be a difficult task.

3. Features for biomedical articles retrieval

A typical medical collection consists of a large number of articles. Here, C denotes the whole collection, $C = \{d_1, d_2, \ldots, d_{|C|}\}$, d_i denotes the document (article) in the collection. $|C|$ denotes the size of the set C. $M = \{m_1, m_2, \ldots, m_{|M|}\}$ denotes a set of query articles posed by the user for article retrieval. $F = \{f_1, f_2, \ldots, f_{|F|}\}$ denotes a set of features used. Formally, a feature f_k is a function which takes into a pair of the query article and the document (M_i, d_j) and outputs a score. We use $f_k(M_i, d_j)$ to denote both the function f_k and the score output by f_k when there is no confusion. Clearly, with any feature f_k, we can get a rank of documents on C for each query article M_i.

3.1. High-level features

This set of features includes most of those features we considered for this paper. The main focus is how to convert a full length query article into an efficient search query. Each high level feature is issued to retrieval function and retrieval score is used as the value of the feature. The queries considered for retrieval function are Indri queries [16]. Indri a well-developed query language, it provides many operators to complete different types of search queries. For example, we can use the field operator to indicate the structure information. It is also easy to provide weights for each word and use the phrase operator.

To extract words from a document, we first sort all the words in the document on the basis of their decreasing word frequencies. Next, we select only those words that have document frequency less than 30% (in the collection) and use these words for generating Indri queries. Precisely, $f_k(M_i, d_j) = f_{ret}(h(M_i), d_j) = f_{ret}(q_i, d_j)$, where f_{ret} is the retrieval function of the search engine and h is a transforming method which generates the effective search query q_i from the query article M_i posed by the user. We used *Indri* as a retrieval function f_{ret}. To transform a article into Indri queries $h(M_i)$, we need to consider several factors: first, whether the query made up of words or other entities; second, where to extract them; third, how to select and assign weights to them; fourth, whether to search the whole article or just some fields.

For the first factor, both words and noun phrases will be considered. As we mentioned before, sometimes a noun-phrase is more useful than the single words that made up the phrase for retrieval. For example, the noun phrase "*network address*" is more helpful than either "*network*" or "*address*". In addition, since a biomedical article is a formal document with correct grammar, current natural language processing techniques can extract noun phrase accurately.

For the second factor, we use six fields of a biomedical articles; the title field *(ttl)*, the abstract *(abst)*, the introduction *(intr)*, the description of the tables and figures *(figr)*, the detailed results *rslt* and discussion *(diss)* and the conclusion *(conl)*. The query items can also be extracted from the whole biomedical article *(all)*. In this case, the structure information will be ignored.

For the third factor, any techniques that measure the importance score of a word or phrase can be used. Here, we only consider some standard statistics like *tf* and *tfidf*. For the fourth factor, the retrieval system can search either the whole biomedical article (all) or specific fields. Here, we consider the six fields with the explicit tags. A general algorithm is provided to transform the biomedical article (query) to an effective search query, where the values of the parameters reflect the four factors mentioned previously. The details of the algorithm can be found in Algorithm 1.

For the general algorithm in Algorithm 1, the possible values of the input parameters are listed in Table 3. Currently, we only list *tfidf* as the possible value of the parameter *Score*, but it is easy to incorporate other values which measure the importance of words or phrases. For the parameter *Weight*, *bool* represents assigning the weight 1 to all items. Here, *tf* is calculated within *Field*.

We can instantiate many transforming methods with different parameter configurations. Table 4 shows some transforming methods and the corresponding queries after applying them to the example query article in Table 1. Since *Indri* is used as the retrieval system, the transformed queries are expressed as the *Indri* query language. Table 4 shows three examples of *Indri* queries for automatically transforming the example query article of Table 1. In *Indri* language, *immunity. (ttl)* means searching "*immunity*" using the language model estimated from the *ttl* field.

Table 2
Summary of ranking techniques used in the related work on retrieving biomedical documents. Last column compares whether or not user was involved in providing initial query keywords.

Related Work	Ranking Technique	User Involved in Providing Initial Query?
[29]	Learning to Rank and Query Expansion	Yes
[9]	Thesaurus based Query Expansion	Yes
[1]	Query Expansion with Pseudo Relevance Feedback	Yes
[11]	Ontology and Clustering	Yes
[13,11,23]	Text Mining using Citation Information	Yes
[22]	Rank Fusion	Yes
[25]	Unsupervised Query Expansion with Pseudo Relevance Feedback	Yes

Table 3
Selected values for the parameters.

Parameter Name	*Values*
Score	*tfidf*
Num	*Integer*
Weight	*tf,idf,bool*
NP	*1,0*
Field	*ttl, abst, intr, figr, rslt, diss, conl, all*

3.2. Low-level features

Indri queries cannot extract some useful information from the query article, therefore, we used two more types of features. The first is low level features. These are features that are mostly used in information retrieval. These are $tf(wordfrequency), idf(inversedocumentfrequency), tfidf$ and their variations. Although these statistics are incorporated into the retrieval function fret used by the high-level features, using these statistics as separate features may bring additional benefit [26,5]. Precisely, $f_k(M_i, d_j) = f_{low}(h(M_i), d_j) = f_{low}(q_i, d_j)$, where q_i is the search query transformed from M_i. $q_i = \{(\delta_1\ w_1), (\delta_2\ w_2), \ldots, (\delta_y\ w_y)\}$, where w_i is the query item and δ_i is the weight of w_i. f_{low} is the equation used to calculate the statistics. In this paper, seven types of equations are used, which are summarized in Table 5.

In Table 5, $c(w_i, d_j)$ represents how many times the w_i appears in d_j. $|d_j|$ is the length of document, $|C|$ denotes the size of collection (total number of documents), and $df(w_i)$ is the number of documents that contain w_i. The architecture of proposed system for automatically transforming full length biomedical article into search query is explained in Fig. 1. The system first extracts high level features and low level features and generates Indri query. The optimal Indri query is generated using learning to rank approach, and then relevant documents are retreived using learning to rank by combining features (explained in Section 4).

Algorithm 1 Algorithm for converting a full length query article to an efficient search query.

Input : *Biomedical Article*, the query *article*; *Field*, the field to extract query items; *Score*, the scores used to measure the importance of the candidate items; *Num*, the number of the extracted query items; *Weight*, the weight assigned for the query items; *NP*, the tag indicating whether use noun phrase; *TField*, the field the system will search

Output: *Query*, an effective search query

Rank all words in *Field* on the basis of their *Score* and then select *Num* high scoring words as keywords for search query. Provide *Weight* to each query word to get the $Query_{word}$;
Check *NP*;
if $NP = 0$ **then**
    $Query \leftarrow Query_{word}$;
else
    repeat Step 1 for noun phrases instead of words to get $Query_{phrase}$. Set *Query* as the combination of $Query_{word}$ and $Query_{phrase}$;
end
Put *TField* into *Query* to tell the system where to search;
Return *Query*;

4. Feature combination

In this section, we consider how to combine different types of features. The main approach used is linear combination due to its simplicity and comprehensibility. Given a set of features $F = \{f_1, f_2, \ldots, f_{|F|}\}$, the new ranking function is $f(M_i, d_j) = \sum_{k=1}^{|F|} \alpha_k f_k(M_i, d_j)$, where α_k is the weight $k = 1$ assigned to the kth feature.

Assigning appropriate values of α_k is essential for the success of this technique. In the situation of related biomedical articles

Table 4
Examples of transforming methods.

	Field	*Sum*	*Num*	*Weight*	*NP*	*TField*
Method1	*ttl*	*tfidf*	*4*	*bool*	*0*	*all*
Query1	*#Weight (1.2 inborn 1.0 chronic 1.1 immunity)*					
Method2	*ttl*	*tfidf*	*4*	*tfidf*	*0*	*ttl*
Query2	*#Weight (3.6 inborn. (ttl) 1.9 immunity. (ttl) 1.6 chronic. (ttl))*					
Method3	*abst*	*tfidf*	*3*	*tf*	*1*	*all*
Query3	*#Weight (0.7 #Weight (1.2 inborn 2.1 immunity 3.0 chronic) 0.2 #Weight (1.1 #2 (inborn human immunity) 1.0 #3 (human immunity underlie) 1.2 #3 (immunity chronic mucocutaneous))*					

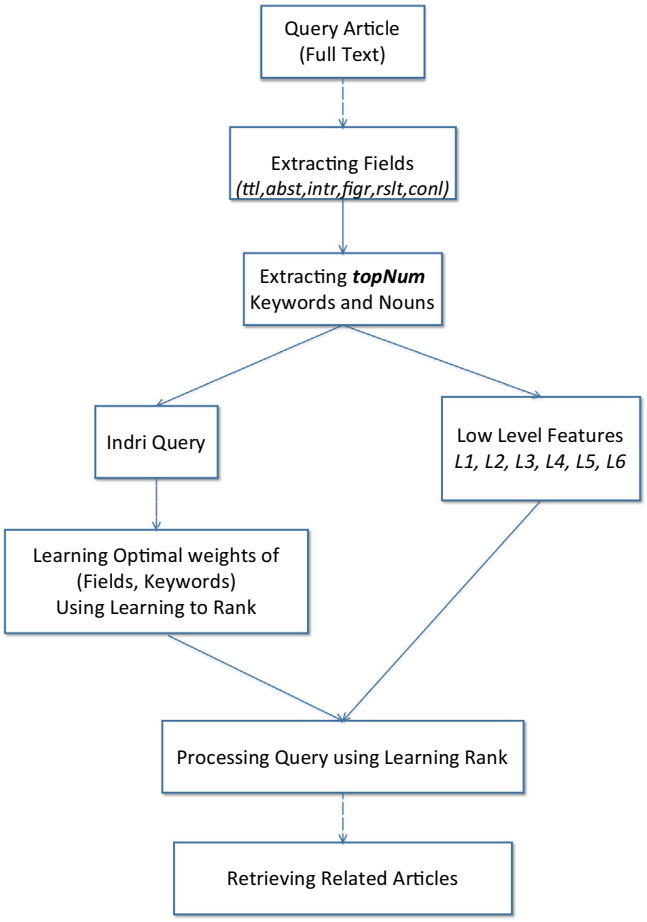

Fig. 1. Architecture of automatically transforming full length biomedical article into search query.

search, the reference field can be used as the relevance judgments for a training set with thousands of queries, $S_{train} = \{(M_1, R_1), (M_2, R_2), \ldots, (P_{|S_{train}|}, R_{|Strain|})\}$, where M_i is the relevant articles for the query M_i. Based on S_{train}, we can easily obtain the performance of each feature on the training set. Since the performance directly reflects the "*strength*" of each feature, it is natural to assign a_k based on the corresponding performance. This approach is called *RankFusion*. Based on the performance measure used, we have *FusionP@10* and *FusionMAP*. Recently, a novel learning to rank method AdaRank [28] was proposed, which learns the combination weights of each feature using a boosting approach.

During each round of training, *AdaRank* computes the effectiveness of all features in the feature set, and selects on features that has the highest effectiveness on the training set. Generally, "*hard queries*" are used for selecting feature when previously selected queries do not provide high effectiveness. For testing the effectiveness on unseen queries, the selected features are combined using learning to rank and documents are ranked using learning to rank. Note that if a feature is not selected during the training phase, the weight of this feature is assigned 0, thus *AdaRank* performs the feature selection implicitly. The details of the *AdaRank* algorithm can be found in [28]. In addition, compared with other learning to rank methods such as *Ranking SVM* [8], *AdaRank* scales well when the training set contains thousands of queries. Similar to *RankFusion*, based on the performance measure used in training phase, we used the two methods *AdaRankP@10* and *AdaRankMAP*.

5. Experiments

In this section, experiments are conducted on a real biomedical articles collection to explore the effect of each feature with different parameter configurations and also demonstrate the performance of the feature combination methods.

5.1. Corpus

We used TREC clinical decision support track for experiments [18]. The target documents is the open access subset of *PubMed* central which contained a total of 733, 138 articles. Our query set consists of 2000 random articles, which have at least 20 citations and all types of fields. From the 2000 documents of query set we randomly select 1500 documents as a training set and 500 documents as a test set.

Relevance judgments are required for queries in order to compare effectiveness of different techniques. Ideally for biomedical and patent retrieval tasks we would need a set of domain experts that could provide us relevance judgments for related articles in the form of ratings by reading all articles. However, it is extremely difficult to obtain relevance judgments from the domain experts as the domain experts need to read large number of articles from the collection. Therefore, in order to provide a reasonable approximation of relevance judgments we use article reference field in the TREC clinical decision support track as a substitute of relevance judgments. *Indri* system is used to index the documents of collection. Six fields with the explicit tags are also indexed in the system. We used the *Porter* stemmer for stemming each word. The standard mean average precision *(MAP)* and precision at *10 (P@10)* are used for analyzing the retrieval effectiveness.

5.2. Effect of single retrieval features

In this subsection, each single feature with different parameter configurations will be explored in order to find the most important factors for related biomedical articles search. For this analysis, it is unnecessary to use the training set to tune the parameters. Therefore, we explore the features on both the training set and the test set. Due to the limit of the space, we only report the performance on the test set, since the results on both sets are very similar.

5.3. Effectiveness of high-level features

For these features, the effectiveness is calculated with following parameters; *Field*, *Num*, *Weight*, *NP* and *TField*. The main challenge that we consider is how many words are required for article search. We test different values for *Num* from *10* to *50* with the gap *10*. The *NP* is set to *0* and the *Weight* is set to *bool*. *TField* is set to "all" (*TField* will take this value in the following experiments, otherwise explicitly stated). Considering the number of words required may differ with different fields, we conduct experiments for all values of *Field*, respectively. The results are shown in Fig. 2 and Fig. 3.

Fig. 2 and Fig. 3 show that using *10* words from the title as the query is enough for article search. With more words, there is almost no change for the performance. This result is reasonable since the titles of most articles are less *10* words. For other fields, the most significant improvement occurs between *10* words and *20* words. When more words are selected, the change in the effectiveness is not significant. It seems that the top *20* words ranked by *tfidf* can capture the most useful information for each type of fields. For the following experiments we select *10* words for the title field *Num*, and *20* words for other fields.

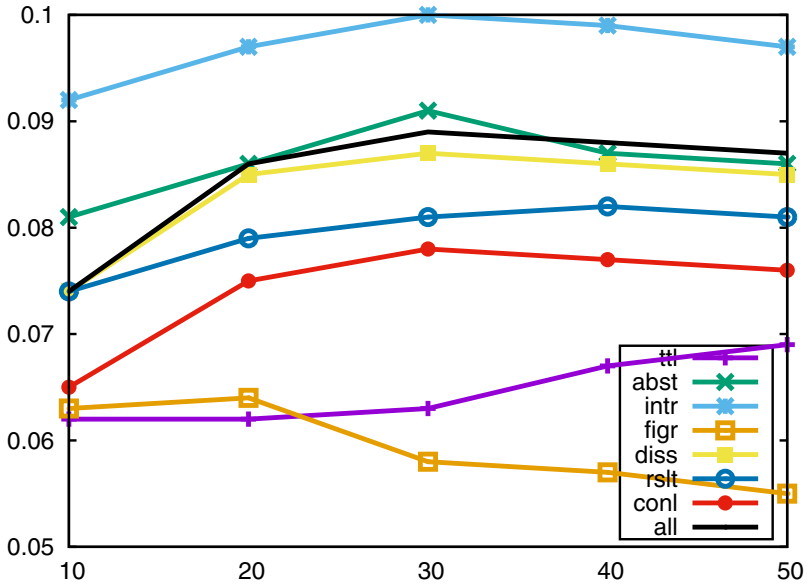

Fig. 2. Performance of *Num* on *MAP*.

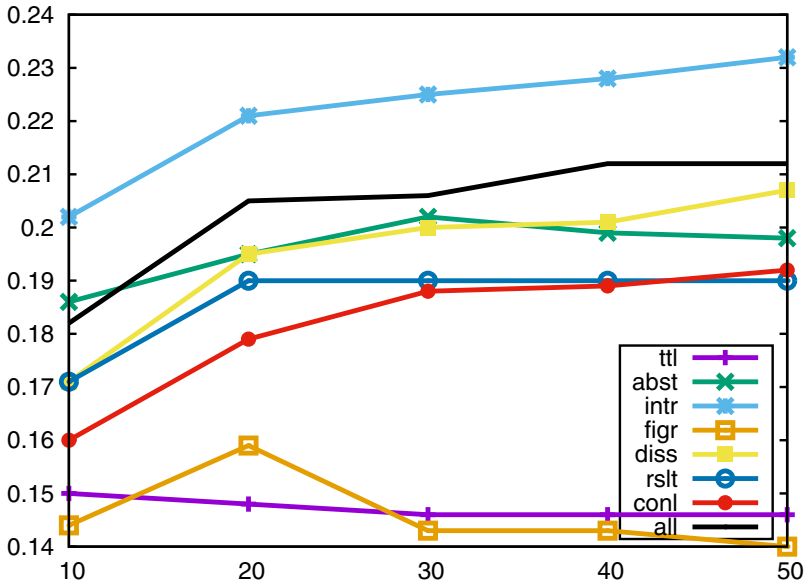

Fig. 3. Performance of *Num* on *P@10*.

The second question we want to answer is which type of weighting methods is the most useful. We explore three values of the parameter *Weight*, i.e. *bool*, *tfidf* and *tf* for different fields. *NP* is also set to *0*. Table 6 shows the results. The results of "*intr*" show significant different with the corresponding values of all other fields. Table 6 shows that *tf* weighting scheme performs better than other weighting scheme. *tfidf* performs better than *bool* but not better as *tf*. Some exceptions happen in the title field and the discussion field. For the title field, it seems that weighting words does not bring much benefit. Since the title usually consists of a small number of words, it makes sense to give all words equal weights. For the discussion field, using *tf* can have some negative effects. In the claim field, as aforementioned, the author tends to use many vague words to extend the coverage of the article. These vague words usually have lower *idf* scores, thus using *tfidf* tends to avoid the influence of those words.

The third question we are interested in is which field is better for extracting the query words. The results of Fig. 2, Fig. 3 and Table 6 show the words extracted from the introduction field provide the best effectiveness. It is interesting to notice that the effectiveness of the introduction field is much better than the discussion field when used for generating queries.

Fourth, we want to explore whether combining words and noun phrases is helpful. Thus, we set *NP* as 1 and *Weight* as *tf*. Table 7 shows the results of each field type. w is when only words are used; $w + p$ is when words and noun phrases are combined with weights 0.7 and 0.3. $*$ shows significance of results. The results of Table 7 shows in most experiments noun phrases improves the effectiveness, however this improvement is not very significant.

The fifth point is about the effect of the parameter *TField*. Until now we only use the structure information to generate the query

Table 5
The functions for calculating scores of low-level features. "nor" is normalized value.

	$f_{low}(q_i, d_j)$	
L1	$\sum_{w_i \in q_i \cap d_j} . \delta_i . c(w_i, d_j)$	*tf*
L2	$\sum_{w_i \in q_i \cap d_j} . \delta_i . \frac{c(w_i, d_j)}{\lvert d_j \rvert}$	*nor(tf)*
L3	$\sum_{w_i \in q_i \cap d_j} . \delta_i . \log(c(w_i, d_j) + 1)$	*log(tf)*
L4	$\sum_{w_i \in q_i \cap d_j} . \delta_i . \log(\frac{\lvert c \rvert}{df(w_i)})$	*idf*
L5	$\sum_{w_i \in q_i \cap d_j} . \delta_i . c(w_i, d_j) . \log(\frac{\lvert c \rvert}{df(w_i)})$	*tfidf*
L6	$\sum_{w_i \in q_i \cap d_j} . \delta_i . \frac{c(w_i, d_j)}{\lvert d_j \rvert} . \log(\frac{\lvert c \rvert}{df(w_i)})$	*nor(tf) idf*
L7	$\sum_{w_i \in q_i \cap d_j} . \delta_i . \log(c(w_i, d_j) + 1) . \log(\frac{\lvert c \rvert}{df(w_i)})$	*log(tf).idf*

words, but not use this information to decide where to search. In other words, given a query, we want to know, compared with searching the whole article, whether it is better to search only the fields corresponding to the source of the query words. Thus, different with our previous setting for *TField ("all")*, now we set it using the value of *Field*. The other parameters are the same as previous, the *Weight* parameter is set to *tf* and parameter *NP* is set to 0. Table 8 shows the retrieval performance. "*all*" denotes searching all content and "*Field*" denotes only searching the field that is the source of the query words. * denotes the performance of "*all*" is significantly different with "*Field*".

Table 8 clearly shows that searching only the corresponding field will significantly decrease the retrieval performance. We also looked in more detail at the effect of the main body *(diss)* field. This field takes up more than half of the content of an article. Taking efficiency into account would mean that if the effect of this field is not obvious, it could be ignored when building the index for the collection. Thus, a new index is built for the collection, where the *diss* field is ignored. We use the following parameters to conduct experiments for this new index, where parameter *Weight* is set to *tf*, and parameter *NP* is set to 0, and parameter *TField* is set to "*all*". Table 9 shows the results. "*with-diss*" denotes the index with all content and "*no-diss*" denotes the index ignoring the *diss* field. * denotes results are significantly different. Table 9 shows that for most cases, indexing the *diss* field is helpful to improve the retrieval performance, especially for *MAP*.

Table 7
Performance of *NP* on retrieval performance.

Field	MAP		P@10	
	w	*w + p*	*w*	*w + p*
ttl	0.063	0.062	0.149	0.154
figr	0.067	0.068	0.160	0.164
conl	0.086	0.086	0.197	0.198
rslt	0.075	0.076	0.179	0.181
diss	0.084	0.086*	0.197	0.200
abst	0.094	0.094	0.211	0.212
all	0.098	0.100*	0.223	0.224
intr	0.114	0.116*	0.239	0.245

Table 8
Influence of *TField* on retrieval performance.

Field	MAP		P@10	
	all	*Field*	*all*	*Field*
ttl	0.063*	0.043	0.149*	0.107
figr	0.068*	0.049	0.161*	0.124
conl	0.087*	0.079	0.197*	0.188
diss	0.085*	0.066	0.197*	0.167
abst	0.095*	0.063	0.211*	0.154
intr	0.114*	0.103	0.240*	0.226

Table 9
Influence of the *diss* field on retrieval performance.

Field	MAP		P@10	
	with-diss	*no-diss*	*with-diss*	*no-diss*
ttl	0.064	0.066*	0.149	0.159*
figr	0.067	0.065	0.161	0.154
conl	0.087*	0.076	0.198*	0.178
rslt	0.075	0.075	0.179	0.179
diss	0.086*	0.082	0.197	0.189
abst	0.095*	0.091	0.210	0.212
all	0.099*	0.095	0.223	0.214
intr	0.115	0.114	0.239	0.245

5.4. Effectiveness of low-level features

The results show that the low-level features do not achieve good retrieval effectiveness. We tried first using a retrieval- score feature to retrieve 1,000 documents and then use the low-level features to re-rank those 1,000 documents. The high-level feature used here is extracting words from the summary field with *tf* weighting because this was the best performance in Table 5. This feature is denoted as *intr*. The low-level features used are *L1-L7* according to Table 6. Fig. 4 shows the performance of *L1-L7* and compares them with *intr*. The results of Fig. 4 show most of low-level features do not improve the effectiveness of *intr*, which is used for retrieving the initial documents of queries. Among the seven features, $L3(log(tf)), L4(idf)$ and $L7(log(tf) * idf)$ show good effectiveness than other features.

5.5. Combining different types of features

In this section, we explore combining different types of features. Although the focus of this article was to retrieve related articles of biomedical domain. However, we also test our approach on a related domain. We select priori art patent retrieval task for this purpose. Prior-art search in patent retrieval is to find previously

Table 6
Performance of *Weight* on retrieval effectiveness.

TField	MAP			P@10		
	bool	*tf*idf*	*tf*	*bool*	*tfidf*	*tf*
ttl	0.061	0.059	0.064	0.138	0.131	0.139
figr	0.063	0.069	0.067	0.148	0.147	0.151
conl	0.076	0.078	0.088*	0.169	0.175	0.191*
rslt	0.079	0.083	0.075	0.179	0.177	0.170
diss	0.086	0.087	0.084	0.185	0.191	0.189
abst	0.086	0.091	0.095*	0.186	0.191	0.200
all	0.087	0.088	0.098*	0.195	0.194	0.212*
intr	0.098	0.102	0.114*	0.211	0.212	0.229*

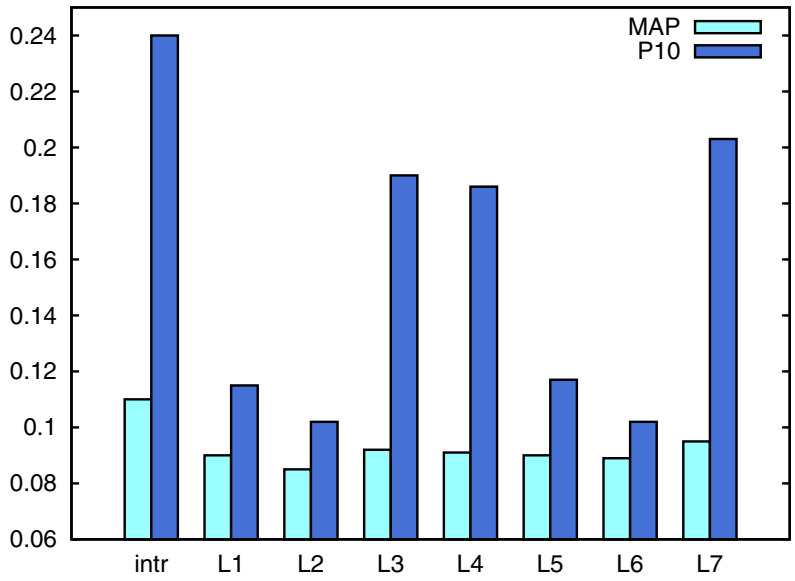

Fig. 4. Retrieval performance of low-level features.

published patents on a given topic. The goal of searching a patent database for the prior art search task is to find all previously published related patents on a given topic [15,7,14]. It is a common task for patent examiners and attorneys to decide whether a new patent application is novel or contains technical conflicts with some already patented invention. They collect all related patents and report them in a search report.

We select prior art (PA) task of the TREC chemical retrieval task (TREC–CRT) for analyzing the effectiveness our approach [14]. The PA task consisted of 1,000 topic queries that are the full-text patent documents (i.e., consisting of at least claims and abstract or description) taken from both the European Patent Office (EPO) and the US Patent Office (USPTO). We randomly split the 1,000 topic queries into the training set with 700 patents and the test set with 300 patents. Similar to bio-medical article, several types of sections are available in this patent. First, the $< TITLE >, < ABST >, < BSUM >, < DRWD >, < DETD >$ and $< CLMS >$ tags indicate the title, the abstract, the summary, the description of the figures, the main body and the claim fields, respectively. We use these sections for generating Indri queries. We use patent citation field of TREC–CRT as a substitute of relevance judgments. Performance results on TREC–CRT collection are shown in Table 12.

The candidate features are shown in Table 10, where *"Ret"* denotes the high-level features, *"Low"* denotes the low-level features. For the noun phrase features, as shown in the third line of Table 10, four fields with the best performance, i.e. *intr*, *abst*, *diss* and *all*, are selected. Two feature sets are compared, one is *Ret* with 42 high-level features and the other is *All* with all 59 features. The best single search feature is denoted as *SingleBest*. In *Ret* and *All*, this is the feature combining words and phrases from the summary fields with *tf* weights. The performance of *FusionMAP*, *FusionP@10*, *AdaRankMAP* and *AdaRankP@10* are reported in Tables 11 and 12 for two collections. ^{s} represents significantly different with the *SingleBest*, and * represent significantly different with the effectiveness on *Ret*.

In order to analyse the effectiveness of our approach we compare the effectiveness of our approach with the keyword based

Table 10
Candidate features.

	Field	*Weight*	*NP*	*TField*	*#*	
Ret	8 fields	bool,tf,tfidf	0	all	24	Table 5
	4 fields	tf	1	all	4	Table 6
	6 fields	tf	0	Field	6	Table 7
	8 fields	tf	0	no-diss	8	Table 7
Low		L1-L7			7	Fig. 3

Table 11
Performance of different combination techniques on TREC clinical decision support task collection. We use the paired *t*-test with significance at $p < 0.05$.

Field	MAP		P@10	
	Ret	*All*	*Ret*	*All*
SingleBest	0.126	0.126	0.256	0.256
FusionMAP	0.130	0.138^{s*}	0.267	0.279^{s*}
FusionP10	0.130	0.137^{s*}	0.268	0.277^{s*}
AdaRankMAP	0.131^{s}	0.138^{s*}	0.268	0.279^{s*}
AdaRankP10	0.133^{s}	0.138^{s*}	0.268^{s*}	0.277^{s*}
Keyword-based Approach with Language Model (JM)		0.088		0.157

Table 12
Performance of different combination techniques on TREC chemical patent retrieval task collection. We use the paired t-test with significance at $p < 0.05$.

Field	MAP		P@10	
	Ret	*All*	*Ret*	*All*
SingleBest	0.096	0.096	0.206	0.206
FusionMAP	0.100	0.108^{s*}	0.218	0.229^{s*}
FusionP10	0.100	0.108^{s*}	0.219	0.226^{s*}
AdaRankMAP	0.102^{s}	0.109^{s*}	0.217	0.230^{s*}
AdaRankP10	0.103^{s}	0.109^{s*}	0.219^{s*}	0.226^{s*}
Keyword-based Approach with Language Model (JM)		0.056		0.112

approach. From each topic in both collections we build the queries by first sorting all the words in the full text on the basis of their increasing word frequencies. Next, we select the top 30 words that have highest frequencies, and use these words in the form of a long query for searching the relevant documents. We process the queries using language modelling approach (Jelinek-Mercer smoothing) with smoothing value of 0.7 [30].

Tables 11 and 12 show after combining different features, performance is significantly better than the best single feature on both collections. Results show combining features also provide better effectiveness than keyword based approach on both collections. Also, the effectiveness on *All* field is significantly better than on *Ret*, which shows that although low-level features and category features are not good when used alone, however they are helpful when used in combination with the high-level features. For different combination techniques, their performance on *P@10* is almost the same no matter which feature set is used. *AdaRank* performs slightly better than *RankFusion* on *MAP* for the *Ret* feature set. It is interesting to compare the number of features selected by *AdaRank* on both feature sets. Among all 42 features of *Ret*, *AdaRankMAP* selects 22 features and *AdaRankP10* selects 17 features. Among all 59 features of *All*, both *AdaRankMAP* and *AdaRankP10* select all 59 features. It seems that the benefit of *AdaRank* mainly comes from its implicit feature selection. The results on the *All* feature set show that when all features are used, the weights learned by *AdaRank* are not substantially different with *RankFusion* methods.

5.6. Discussion

It should be noted that the highest performance achieved in the above experiments *(MAP: 0.128, P@10: 0.249)* for TREC decision support system task and *(MAP: 0.109, P@10: 0.230)* for TREC–CRT are still lower than other typical search applications. The main reason for this is the relevance judgments used from the biomedical and prior-art patent retrieval domains. The following analysis shows the limit of using citation articles as relevance judgments.

First, some relevant articles missed by our system are actually only marginally related to the query article. For example, given the query article with the title "*Iron deficiency anemia*", the relevant article indicated by the reference field include "*Individualized treatment for iron-deficiency anemia in adults*" and "*Pallor in diagnosis of iron deficiency in children*". Second, some top ranked articles returned by our system appear to be relevant but are missing in the reference field. For example, given the query article with the title "*Risk of primary infection and reinfection with respiratory syncytial virus*", our system returns "*Respiratory Syncytial Virus*" and "*Respiratory syncytial virus disease in infants despite prior administration of antigenic inactivated vaccine*". However, both of them are missing in the reference field. Part of reason for this lies that no matter how careful an article author is, it is almost impossible for him or her to find all related articles.

6. Conclusion

Retrieving related articles from medical resources is an important task in the biomedical information retrieval. Previous research on biomedical information retrieval is typical based on keyword search. In this paper we consider a new approach. Using our approach, a user can directly use the whole medial article as a query instead of just the query words. This reduces the burden on the users and generates an effective automatic query from many more useful search features. Given an article as a query, the system utilizes the full abundant information available in the article and searches many more potentially useful retrieval features such as the high-level features and the low-level features. On the *PubMed* collection, we explored each single feature with different parameter configurations, as well as combinations of these features using the techniques of learning to rank and rank fusion. The best single feature found is to combine the words and noun phrases from the summary field with word frequency as the weighting method. Also, using rank fusion and learning to rank methods to combine features can significantly improve the performance relative to the best single feature.

In the future, obtaining more reliable relevance judgments is the most important issue. Other techniques for extracting concepts and entities from articles text should be further explored, as should the potential of the category feature. Current research on biomedical information retrieval does not help physicians for predicting the quality of queries. An interesting future work is to design a system that help the physicians on predicting the quality of their queries. This would help the physicians to take remedial actions in advance in case of bad quality queries without spending enormous amount of time on reformulating search queries. Previous research on query quality prediction relies only on the keywords of given query. This is suitable in case of web retrieval where retrieval systems do not any have additional information about the search topic. However, in case of biomedical information retrieval a rich source of information is available about the search topic in the form of full text of article. This additional information can be utilized together with the initial query's keywords for increasing the quality of prediction.

References

[1] Abdou S, Savoy J. Searching in medline: Query expansion and manual indexing evaluation, vol. 44. Tarrytown, NY: USA. Pergamon Press Inc.; 2008. p. 781–9.
[2] Alsulmi M, Carterette B, Improving medical search tasks using learning to rank. In: 2018 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology, CIBCB 2018, Saint Louis, MO, USA, May 30 - June 2, 2018, pages 1–8.
[3] Burges C, Shaked T, Renshaw E, Lazier A, Deeds M, Hamilton N, Hullender G. Learning to rank using gradient descent. Proceedings of the 22Nd International Conference on Machine Learning, ICML '05. New York, NY, USA: ACM; 2005. p. 89–96.
[4] Burke DT, DeVito MC, Schneider JC, Julien S, Judelson AL. Reading habits of physical medicine and rehabilitation resident physicians. Am J Phys Med Rehabil 2004;83:551–9.

[5] Cummins R, O'Riordan C. Learning in a pairwise term-term proximity framework for information retrieval. In: SIGIR ' 09: Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY, USA: ACM; 2009. p. 251–8.
[6] Freund Y, Iyer R, Schapire RE, Singer Y, An efficient boosting algorithm for combining preferences. volume 4, pages 933–969. JMLR.org; 2003.
[7] Fujii A. Enhancing patent retrieval by citation analysis. In: SIGIR ' 07: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM; 2007. p. 793–4.
[8] Herbrich R, Graepel T, Obermayer K. Advances in large margin classifiers. MIT Press; 2000. p. 115–32.
[9] Hersh WR, Price S, Donohoe L. Assessing thesaurus-based query expansion using the umls metathesaurus. AMIA Symposium. American Medical Informatics Association.
[10] Lillis D, Toolan F, Collier R, Dunnion J. A probabilistic approach to data fusion. Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '06. New York, NY, USA: ACM; 2006. p. 139–46.
[11] Lin Y, Li W, Chen K, Liu Y. A document clustering and ranking system for exploring medline citations. J Am Med Inform Assoc 2007;14:651–61.
[12] Lu Z, Pubmed and beyond: a survey of web tools for searching biomedical literature. In Database, volume 11; 2011.
[13] Lu Z, Kim W, Wilbur WJ. Evaluating relevance ranking strategies for medline retrieval. J Am Med Inform Assoc 2009;16:32–6.
[14] Lupu M, Huang J, Zhu J, Tait J. TREC-CHEM: large scale chemical information retrieval evaluation at trec. SIGIR Forum, ACM 2009;43(2):63–70.
[15] Mase H, Matsubayashi T, Ogawa Y, Iwayama M, Oshio T. Proposal of two-stage patent retrieval method considering the claim structure. ACM Transactions on Asian Language Information Processing (TALIP) 2005;4(2):190–206.
[16] Metzler D, Strohman T, Turtle HR, Croft WB, Indri at trec 2004. In Terabyte track, TREC; 2004.
[17] Murphy LS, Reinsch S, Najm WI, Dickerson VM, Seffinger MA, Adams A, Mishra SI, Searching biomedical databases on complementary medicine: the use of controlled vocabulary among authors, indexers and investigators. In BMC Complementary and Alternative Medicine, volume 3; 2003.
[18] Palotti J, Hanbury A, Tuw @ trec clinical decision support track 2015. In TREC; 2015.
[19] Palotti J, Hanbury A, Müller H, Kahn Jr CE. How users search and what they search for in the medical domain, vol. 19. Hingham, MA, USA: Kluwer Academic Publishers; 2016. p. 189–224.
[20] Roberts K, Simpson MS, Demner-Fushman D, Voorhees EM, Hersh WR. State-of-the-art in biomedical literature retrieval for clinical cases: a survey of the TREC 2014 CDS track. Inform Retrieval J 2016;19(1–2):113–48.
[21] Roy B. The outranking approach and the foundations of electre methods. Theory Decision 1991;31:49–73.
[22] Shaw JA, Fox EA, Shaw JA, Fox EA, Combination of multiple searches. In The Second Text Retrieval Conference (TREC-2), 243–252; 1994.
[23] Siadaty MS, Shu J, Knaus WA, Relemed: sentence-level search engine with relevance score for the medline database of biomedical articles. In BMC Medical Informatics and Decision Making, volume 7; 2007.
[24] Sneiderman CA, Demner-Fushman D, Fiszman M, Ide NC, Rindflesch TC. Knowledge-based methods to help clinicians find answers in medline. J Am Med Inform Assoc 2007;14:772–80.
[25] Soldaini L, Yates A, Goharian N. Learning to reformulate long queries for clinical decision support. JASIST 2017;68(11):2602–19.
[26] Tao T, Zhai C. An exploration of proximity measures in information retrieval. In: SIGIR ' 07: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY, USA: ACM; 2007. p. 295–302.
[27] Xu B, Lin H, Lin Y, Ma Y, Yang L, Wang J, Yang Z. Improve biomedical information retrieval using modified learning to rank methods. IEEE/ACM Trans Comput Biol Bioinform (TCBB) 2018;15(6):1797–809.
[28] Xu J, Li H. Adarank: A boosting algorithm for information retrieval. Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '07. New York, NY, USA: ACM; 2007. p. 391–8.
[29] Yu H, Kim T, Oh J, Ko I, Kim S. Refmed: Relevance feedback retrieval system fo pubmed. Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM '09. New York, NY, USA: ACM; 2009. p. 2099–100.
[30] Zhai C. Risk minimization and language modeling in text retrieval. Carnegie Mellon University; 2002. PhD Thesis.
[31] Zhang Y, Searching for specific health-related information in medlineplus: Behavioral patterns and user experience. volume 65, 53–68; 2014.
[32] Zuccon G, Koopman B, Palotti J. Diagnose this if you can. In: Hanbury A, Kazai G, Rauber A, Fuhr N, editors. Advances in Information Retrieval: 37th European Conference on IR Research, ECIR 2015, Vienna, Austria, March 29 - April 2, 2015. Proceedings. Cham: Springer International Publishing; 2015. p. 562–7.