Supporting BioMedical Information Retrieval: The BioTracer Approach*

Heri Ramampiaro[1] and Chen Li[2]

[1] Department of Computer and Information Science
Norwegian University of Science and Technology (NTNU)
N-7491, Trondheim, Norway
heri@idi.ntnu.no
[2] Dept. of Computer Science
University of California, Irvine (UCI)
Irvine, CA 92697-3425, USA
chenli@ics.uci.edu

Abstract. The large amount and diversity of available biomedical information has put a high demand on existing search systems. Such a tool should be able to not only retrieve the sought information, but also filter out irrelevant documents, while giving the relevant ones the highest ranking. Focusing on biomedical information, this work investigates how to improve the ability for a system to find and rank relevant documents. To achieve this goal, we apply a series of information retrieval techniques to search in biomedical information and combine them in an optimal manner. These techniques include extending and using well-established information retrieval (IR) similarity models such as the Vector Space Model (VSM) and BM25 and their underlying scoring schemes. The techniques also allow users to affect the ranking according to their view of relevance. The techniques have been implemented and tested in a proof-of-concept prototype called BioTracer, which extends a Java-based open source search engine library. The results from our experiments using the TREC 2004 Genomic Track collection are promising. Our investigation have also revealed that involving the user in the search process will indeed have positive effects on the ranking of search results, and that the approaches used in BioTracer can be used to meet the user's information needs.

Keywords: Biomedical Information Retrieval, Evaluation, BioTracer.

1 Background and Motivation

The continuous increase in the amount of available biomedical information has resulted in a higher demand on biomedical information retrieval (IR) systems. While their use has helped researchers in the field to stay updated on recent literature, many of the existing search systems tend to be either too restrictive (returning results with a low recall) or too broad (finding results with a

* This article is a revised and an extended version of the ITBAM 2010 paper [1].

A. Hameurlain et al. (Eds.): TLDKS IV, LNCS 6990, pp. 73–94, 2011.

low precision). For this reason, there is a need to improve existing search systems, especially with respect to retrieval performance, in order to improve their precision and recall.

1.1 Challenges for Biomedical Information Retrieval

From the information retrieval point of view, there are many challenges in retrieving biomedical information. First, as in most scientific domains, there is a wide use of domain-specific terminology [2]. For example, most of the documents extensively use specific gene names, names of diseases and/or specific biomedical specific terms as part of the text. Most methods should take this into account to be able to successfully retrieve relevant information. As a result, it is challenging to offer a unified method for preprocessing, indexing, and retrieving biomedical information.

Second, the mixture of natural English terms and biomedical-specific terms can pose problems due to high term ambiguity. A single word can have different meanings [3]. This ambiguity may in turn result in challenges for traditional IR methods, such as thesaurus-based extension of queries, as well as identifying relevant documents. For this reason, there is a strong need for word sense disambiguation, which is not easy, and is in itself an area of active research [3]. To illustrate, the term "*SOS*" normally refers to "*urgent appeal for help*", but it could also mean the gene symbol "*SOS*", short for "*Son of sevenless*"[1]. Another term illustrating this challenge is "*NOT*". It can refer to a protein[2]. But in traditional IR, it would be categorized as a stop word and, thus might be ignored in the indexing process.

Third, one of the problems with biomedical information is the lack of widely recognized terminology standards. New names and terms are created every time a biologist discovers a new gene or other important biological entities, and there often exist several inconsistent typographical/lexical variants [2]. Authors also normally have their own writing style [4], which can further worsen the situation.

Fourth, partly as a result of the existence of several term variants, important words and symbols suitable for indexing have often a low occurrence frequency, and many of terms appear only once in the entire document collection. While this may, in some cases, be useful since the discrimination effectiveness is often inversely proportional with the word's document frequency, it may also mean a high data sparseness, which would, in turn, have negative effects on the retrieval performance [5].

In summary, current biomedical IR systems have to deal with heterogeneous and inconsistent information. For static document collections, the above characteristic would be less problematic. However, document collections can be very dynamic. As shown in Figure 1[3], every day approximately 1000 new citation

[1] See for example http://www.ncbi.nlm.nih.gov/sites/entrez?db=gene&cmd=retrieve&dopt=default&rn=1&list_uids=6654

[2] See for example http://www.uniprot.org/uniprot/P06102

[3] This graph was generated based on the data at http://www.nlm.nih.gov/bsd/licensee/baselinestats.html

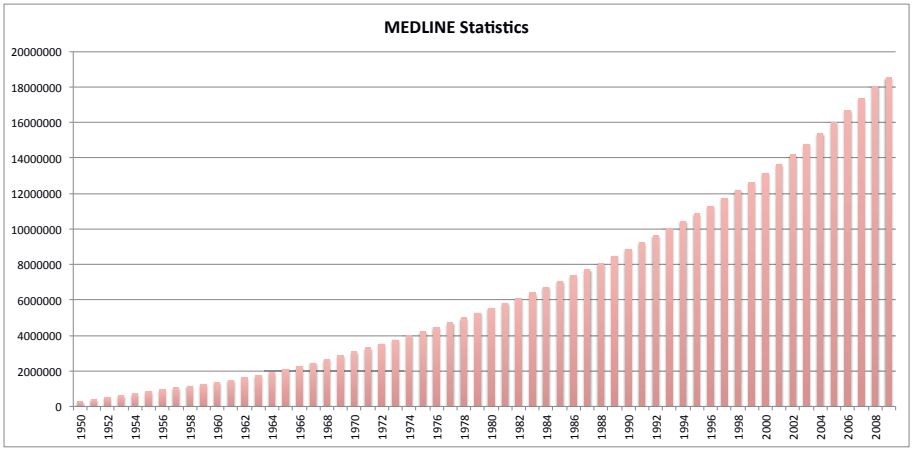

Fig. 1. Number of MEDLINE citations from 1950 to 2009

records are added to MEDLINE[4]. In addition, there is a challenge to measure document relevance to queries. While the notion of relevance based on, for example, the vector space model [6, 5, 7] is well-known and works very well for traditional IR, it needs improvements in biomedical IR, mainly because of the characteristics described above. As a result, the notion of relevance must be examined carefully, knowing that for a specific query, what is relevant for one biologist may not necessarily be relevant for another.

1.2 Objectives and Contributions

Our goal is to provide a generic biomedical IR system that can be *adapted* to meet different information needs. A basic requirement for such a system is that it must address the fact that users often have their own view of relevance. Moreover, the heterogeneity of biomedical information has made it important to have a customized system. This means a system that

(1) allows a user to specify relevant documents from the returned search results through user relevant feedback (URF), and
(2) allows us to specify the similarity/language model and the importance of specific parts in a document.

From this point of view, the main objective of this work is to investigate information retrieval methods that provide the best ranking of relevant results. More specifically, the main contribution of this work is improving existing typically keyword and parametric-based similarity models, such as the BM25 model,

[4] MEDLINE is a trade-mark of the U.S. National Library of Medicine (See http://www.nlm.nih.gov/databases/databases_medline.html)

by enabling boolean queries and wildcard queries, and by applying a query-document-correlation factor to affect the ranking with respect to the amount of overlap between a query and a document (see also Section 3). We provide an extensive evaluation of our method based on known evaluation principles and a proof-of-concept prototype implementing the method. Although we recognize its usefulness, we show that we can achieve the improvement without using a controlled vocabulary such as the Unified Medical Language System (UMLS)[5] and the Medical Subject Headings (MeSH) [8].

1.3 Paper Outline

The remainder of this paper is organized as follows. In Section 2 we discuss what text operations can be used in preparation of index terms. In Section 3, we elaborate on our approach to find optimal scoring schemes and discuss the effects of applying boolean operations with BM25. In Section 4 we discuss how users can be involved in ranking of search results, through user relevant feedback and a fast, interactive user interface. Then, in Section 5 we present and discuss the results from testing the method and the prototype. Section 6 discusses related work. Finally, in Section 7 we summarize the paper and present future work.

2 Choosing Optimal Text Operations

Applying text operations in information retrieval is a process of choosing candidate words or tokens to index [5, 7]. Tokenization and stemming are examples of such operations. While tokenization is straightforward for general English text, this is not necessarily the case with biomedical information due to the characteristics described in Section 1.1. Existing empirical studies on tokenizing biomedical texts further emphasize this [9, 10].

Both studies suggest applying *conditional stemming* as part of the normalization process after tokenization. Stemming means that we only index the root or the *stem* of a specific term, while at the same time being able to search all grammatical variants of that term – e.g., `activate` versus variants such `activates`, `activated`, and `activating`. Based on the results from [9, 10], we believe that stemming would work if it is customized to biomedical texts. However, because the benefit versus cost of stemming is often debatable [5, 7], we chose to compare the effects of the use against omitting stemming. Our conclusion is that stemming did not give significant benefits in terms of retrieval performance. Further, for some types of queries it could change the semantic meaning of a query. For example, a query to find all proteins that are *activated* by a specific protein could also include a query finding all proteins that *activate* the actual protein. This would clearly be a different biological process. With conditional stemming we could take such semantic differences into account. However, this is beyond the scope of this work.

[5] See http://www.nlm.nih.gov/research/umls/index.html

So instead of stemming, we considered the use of *wildcard queries*. With a wildcard query, we can choose a specific query term as a prefix. Then, we extend the original query with all terms in the index that have the same prefix. For example, a query with `ferroportin*` will find articles with `ferroportin1` or `ferroportin-1`. This approach gives the users the freedom to choose when to extend the queries with the term variants. Our experiments also show its usefulness in terms of better retrieval performance.

We also investigated the use of stop-word removal as part of the text operations. This process removes words or terms that do not contribute to discriminating documents. Our experiments show that by carefully choosing candidate words in this process, we can achieve good results at the same time reduce the size of the index.

3 Finding Optimal Scoring Scheme towards Optimal Document Rankings

To further address some of the challenges in Section 1, we need to develop a good ranking method. Existing scoring schemes are good for general purposes, but they need improvements to meet our needs in biomedical information. When developing our method and prototype, we investigated several different schemes, and used them as a starting point towards a more improved scheme. In addition to designating suitable ranking models, we also try to find out how we can influence the ranking by boosting important parts in the document and constraining documents based on their query matching degree.

3.1 Choosing Ranking Models

Several ranking or similarity models have been proposed over the years. One of the earliest, proposed by Salton and Buckley [6], was the Vector Space Model (VSM). It is a similarity model based on cosine similarity between a specific query and the documents in the collection. Its main idea is that for all documents in the collection that contain the terms in the query, a ranking is produced by measuring the cosine of the angle between the query and the document vectors. For general text, it has been shown that this model gives good retrieval performance [5, 11]. Because VSM is one of the most used models, we applied it as a baseline in our experiments. In addition, we implemented an extension based on Lucene[6] [12] .

Another similarity model that has been increasingly popular in information retrieval is the Okapi BM25 [13]. In contrast to the VSM, BM25 is a probability-based approach. As with VSM, it has been shown to be effective on general text retrieval [7]. This is also partly why we decided to implement it in this work. Moreover, we want to know how it performs when used in a highly scientific domain such as the biomedical domain. We first implemented the baseline model

[6] See http://lucene.apache.org/java/docs

as an extension to Lucene. Then, as shown below, we extended it to take into account that parts of the documents have different boosts, which will, as a result, also affect the way the search results are ranked.

Furthermore, in order to compare their performance in the biomedical domain, we implemented Lemur TF-IDF [14] and Divergence From Randomness BM25 (DFR-BM25) [15]. These are models developed as an alternative to the Okapi BM25 model [15, 14, 13]. Lemur TF-IDF is a variant of TF-IDF using a TF component that is similar to the one used in BM25. The Divergence From Randomness (DFR) BM25, on the other hand, is a probabilistic model in the same category as BM25. However, while BM25 uses some parameters to experimentally tune the model, DFR BM25 is non-parametric. As can be derived from the name, it assumes that the distribution of words in documents is a random process and that documents can be ranked by the probability that the actual term distribution found in a document would appear by chance.

Table 1. Summary of considered ranking models

Models	Description
VSM	A vector space model based on TF-IDF weighting scheme.
Okapi BM25	A parametric probabilistic model using TF, IDF, and document length.
Lemur TF	A variant of TF-IDF that uses the same TF parameters as in BM25.
DFR BM25	A non-parametric model using term probability distributions as a base for the result ranking.
Extended VSM	Extending the standard VSM model with query overlap factors, boolean operators, and wildcard queries.
Extended BM25	Extending the standard BM25 model with query overlap factors, boolean operators, and wildcard queries.

In addition to the above models, we have considered using language models (LM) for information retrieval [16, 11]. The basic idea of the language model is similar to the probabilistic model. However, instead of modelling the probability for a document to be relevant as with a probabilistic model, in a language model we build a probabilistic model for each document in our collection and rank our documents with respect to the probability of the model to generate the query. For this reason, this model would fit well with our goals on ranking of documents based on the user' s interactions. Studies have also shown that the LM competes well with the other models with respect to retrieval performance [16, 11]. Nevertheless, due to time constraints its use and integration with BioTracer have been left for future studies.

3.2 Boosting Specific Document Parts

Boosting specific parts in a document is one way to specify how important one part in a document is compared to other parts. By defining this importance,

we implicitly specify how the documents in the search results should be ranked. A document part can be (1) a structural part such as the title or the abstract, or (2) specific sentences or terms. Previous studies have shown that boosting structural parts of documents can improve the result quality [17, 18].

Structure-Based Boosting: By default, we perform search on both the *title* and the *abstract* fields of a document. These fields can be given boosts during index time or search time. A title can describe the content of a document well. Thus terms in a title can readily be assumed important, and given a higher weight than those in other places in the document. For instance, our experiments show that weighting the *title* field twice as much as *abstract* yields best ranking. Another advantage of this approach is that by having different weights for these fields, we also account for the fact that many MEDLINE documents may only contain a title and no abstract. With the same weight, documents without an abstract would automatically be ranked lower.

Sentence and Term-Based Boosting: We also investigated the effect of boosting sentences and terms during the index time. This has several advantages. First, it allows us to systematically specify which sentences or terms are to be weighted higher or lower than others. Second, specifying higher boosts or weights based on the importance of specific terms and sentences can help optimizing the ranking of retrieved documents.

The idea is to use techniques from the text mining and natural language processing (NLP) field, including named-entity recognition (NER) [19, 20] to identify special sentences and words. Such sentences could be those containing biological-specific names such as drug, gene and/or protein names, or biological processes such as protein-to-protein interaction and DNA evolution. The idea of identifying specific words or entity in texts is still an active research area [19, 20, 21] because of the diversity in biomedical text as discussed in Section 1.1. To our best knowledge, there has not been much work that has addressed boosting sentences and words as part of a ranking strategy. By successfully identifying biomedical specific words and sentences we can weight these higher than other general words and sentences. These weights will, in turn, contribute to ranking specific documents higher than other documents.

Our preliminary experiments in this area have shown promising results [22]. It is still in an early stage and for this reason, more work and experiments are needed to make it an integrated part of the current work.

Query-Time Term Boosting: In addition to index time boosting, specific terms can be given higher boosts than other terms during search time. The main advantage of this method is that terms that a user considers to be more important can be given an additional weight, in addition to their statistical information, such as term frequency (tf) and document frequency (df). As an example, assume we want to find all articles describing the function of the protein `ferroportin1`. We could compose a query as `ferroportin1^2.0 protein`

`function`. This query tells the search engine that we would like to weight `ferroportin1` as twice as much as the words `protein` and `function`.

3.3 Constraining Documents Based on Their Query Matching Degree

Computing term weights: In general, a document can be given a score based on its similarity to the query:

$$Score(q, d_j) = sim(q, d_j) = \sum_{i \in q} w_{ij} \cdot w_{iq}. \tag{1}$$

Here $\overrightarrow{d_j} = (w_{1j}, w_{2j}, \ldots, w_{sj}), w_{ij} \geq 0$ is the vector of term weights for a document d_j, and $\overrightarrow{q} = (w_{1q}, w_{2q}, \ldots, w_{sq}), w_{iq} \geq 0$ is a query q, where s is the number of terms in the index.

Now, assume d_j consists of n fields f_{kj}, $k = 1, 2, \ldots, n$. Taking the boosting values into account, we have

$$\overrightarrow{d_j^*} = \sum_{k=1}^{n} \beta_k \cdot \overrightarrow{f_{kj}}, \tag{2}$$

where β_k denotes the boosting value for field f_{kj}. A special case can be defined as

$$\overrightarrow{d_j^*} = 2.0 \cdot \overrightarrow{f_{1j}} + 1.0 \cdot \overrightarrow{f_{2j}},$$

where f_{1j} and f_{2j} are the title and abstract fields, respectively.

Using the Okapi BM25 model [13], we assign each term in d_j and q the following weights w_{ij} and w_{iq}, respectively:

$$w_{ij} = \log\left(\frac{N - df_i + 0.5}{df_i + 0.5}\right) \cdot \frac{(k_1 + 1)tf_{ij}}{K + tf_{ij}} \text{ and } w_{iq} = \frac{(k_3 + 1)tf_{iq}}{k_3 + tf_{iq}}, \tag{3}$$

where $K = k_1((1 - b) + b(L_{d_j}/L_{avg}))$, N is the number of documents in the collection, df_i is the document frequency – i.e., the number documents in the collection that a term t_i occurs in, tf_{ij} is the frequency of a term t_i within d_j, L_{d_j} is the length of d_j – i.e., the number of terms in d_j, L_{avg} is the average document length, while k_1, k_3 and b are tuning constants with default values 1.2, 2.0. 0.75, respectively[7].

Now, let w_{ikj} be the weight of a term in f_{ij} of d_j^*. Then, applying weight on Eq. 3, we have [18]:

$$w_{ikj} = \log\left(\frac{N - \mathrm{df}_t + 0.5}{\mathrm{df}_t + 0.5}\right) \cdot \frac{(k_1 + 1)\beta_k \mathrm{tf}_{ij}}{K + \beta_k \mathrm{tf}_{ij}}, \tag{4}$$

where $K = k_1((1 - b) + b(L_{d_j}/L_{avg}))$.

[7] The values were set in accordance with the recommendation by Robertson and Jones [13].

Query-Document Correlation Factor: In the attempt to improve the ranking of relevant documents, we have added a new factor based on the degree of query matching. The idea is to add a scoring factor based on how well a document matches a query. We call this constraining documents based on their query matching degree, or simply *query-document correlation factor*. Such a constraining process is used in the literature. For instance, Lucene allows a form of document constraining too [12]. In our approach, however, we apply a slightly modified factor. In addition, we combine it with the similarity model BM25 [13]. To our best knowledge, using the idea of document query correlation factor with BM25 is new. Note that this factor does not directly help to increase the overall retrieval precision or recall, but it is meant to influence the ranking of already retrieved results. Its main goal is to maximize the number of relevant documents in the top-k hits of the search results.

Inspired by the scoring scheme in Lucene [12], we extended the original BM25 scoring equation by adding a document-query correlation factor, called $\Gamma(q, d_j)$, for a query q and a document d_j. This factor is calculated based on how many terms in q are found in d_j, and is proportional to the overlap between q and d_j. In our approach, $\Gamma(q, d_j)$ is computed as follows:

$$\Gamma(q, d_j) = \left(\frac{n_{over}(q, d_j)}{max_l \; n_{over}(q, d_l)} \right)^{\theta} = \left(\frac{\sum_{i=1}^{n} n_{over}(q, f_{ij})}{max_l \; n_{over}(q, d_l)} \right)^{\theta} \tag{5}$$

Here θ is a natural number, $n_{over}(q, d_j)$ is the number of terms in q that overlaps with document d_j. For all documents in the result set, $max_l \; n_{over}(q, d_l)$ is the maximum n_{over} for the retrieved documents.

Formally, $n_{over}(q, d_j)$ is defined as follows. Assume that t_i is a term index, $i = 0, 1, \ldots, s$, and s is the total number of terms in the index, i.e., $S = \{t_0, t_1, \ldots, t_s\}$ is the set of all distinct terms in the index. The overlap between a document d_j and q is defined as $\forall t \in S \mid t \in d_j \bigcap q$, $n_{over}(q, d_j) = |d_j \bigcap q|$.

We studied the effect of the factor Γ on the retrieval results and the overall evaluation result. We observed that with a small value of θ, the factor is too dominating. Thus it has unwanted effects on the ranking – i.e., relevant documents come too late in the result lists. Similarly, with a big value, θ will be too restrictive and the factor will have bad influence on the ranking. Therefore, the value of θ has to be chosen carefully. After several experiments, we found that $\theta = 4$ gave optimal results.

By combining Eq. 2, Eq. 4, and the extension of the similarity score from Eq. 1, we have:

$$Score(q, d_j) = \Gamma(q, d_j) \cdot sim(q, d_j^*) = \Gamma(q, d_j) \cdot \sum_{\forall i \; \mid \; t_i \in q} w_{ikj} \cdot w_{iq}. \tag{6}$$

3.4 Applying Boolean Queries with BM25

As for the Vector Space Model [6], BM25 was originally designed for keyword-based queries [13]. However, to be able to restrict the search results and filter out unwanted hits, we wanted to investigate the effect of using boolean operations in

combination with BM25. Therefore, we implemented our prototype to allow users to search with boolean operations including AND, OR and NOT in combination with BM25.

The boolean query model was used in many search engines [5,7]. However, the original model has been seen to be too limited for several reasons. First, it does not rank retrieved results based on relevance to the query, and it only retrieves results that exactly match the query. To address this problem, we investigated the effects of combining boolean queries with BM25. The benefit is that, we can filter the search results based on the boolean operations in the query, and at the same time rank them based on the BM25 model. Second, a general criticism against boolean query usage is that users might be unwilling or even lacking the ability to compose this type of queries. While we recognize that this concern is valid, we still believe that a system has to provide the possibility of processing boolean queries. In fact, a PUBMED query log [23] with approximately 3 million queries shows that 36.5% of the queries were boolean queries. Although many of these queries might be computer generated (e.g. by automatic query extension), it is a strong enough motivation for investigating the effects of boolean queries. Moreover, although boolean queries do pose some challenges when it comes to finding all relevant documents from the collections, our experiments (see also Section 5.2) show that we can improve the average top-100 precision by 17% compared to the baseline BM25. This result means that boolean queries combined with BM25 can have good effect on the search precision.

4 Other Features

4.1 User Relevant Feedback (URF)

As suggested in our previous discussion, there are many ways to involve a user in influencing the ranking of documents. Involving the user through relevant feedback has been studied within traditional and multimedia information retrieval for decades. However, within the biomedical domain it seems still missing. A challenge here is to choose the right strategy. In our work, we have chosen to investigate the use of URF based on the scoring scheme suggested by Robertson and Sparck Jones [13]. In addition, we apply the extended BM25 as described earlier. That is, we also extended the original model with the document-query similarity factor $\Gamma(q,d)$. The scoring scheme equation is similar to the one in Eq. 6, but now the term weight includes the relevance information added by the users. Based on the ideas of Robertson and Sparck Jones [13] and Eq. 4, this weight can be expressed by

$$w_{ikj} = \log\left(\frac{(r+0.5)(N-n-R+r+0.5))}{((n-r+0.5)(R-r+0.5))}\right) \cdot \frac{(k_1+1)\beta_k \mathrm{tf}_{ij}}{K+\beta_k \mathrm{tf}_{ij}}, \tag{7}$$

where $K = k_1((1-b)+b(L_d/L_{avg}))$, r is the number of retrieved relevant documents, and R is the number of relevant documents in the collection for a specific query.

Here, the value of R can be derived from users' click-through information in the query log. Click-through information may work as relevance judgement information in that we can assume that all documents that a user has clicked are relevant to a specific query [7]. This means that we can estimate the number of relevant documents based on the number of documents that are known relevant to the user. Although R may in this case be much smaller than the real total number of relevant documents in the collection, this estimation can still be reasonable [13], presuming that these R relevant documents are within a subset of the set of all relevant documents.

To be able to evaluate this approach, we simulated the user relevant feedback process using the relevance information from the test collection.

4.2 Interactive Autocompletion

The primary goal of a search system is to provide best possible retrieval performance in terms of good search results such as high recall and precision values. However, we argue that providing a tool that is able to provide results within a reasonable time frame is crucial. Users often require an interactive system when performing their searches. For this reason, keeping the search time as low as possible should be an important goal for every retrieval system.

As a result, we studied how to make our system interactive. Our system has the following features. First, BioTracer provides a responsive and interactive autocompletion, which is activated by each keystroke. This feature can help users compose their queries. Furthermore, as opposed to autocompletion in existing systems, autocompletion in BioTracer not only provides possible prefix-based suggestions, but also is fault-tolerant. This means that it can find answers even if the prefix in a query has spelling mistakes. Figure 2 illustrates how this feature is implemented in BioTracer.

These features are useful since quite often a user only vaguely knows what she/he is looking for. Thus by allowing interactive browsing, we can allow a user to decide which suggested keywords should be the correct ones. To enable this feature, we implemented the system as follows (see also Figure 3).

Index term extraction. We use the index term dictionary to build the autocompletion index. To do this, we have to first extract all unique terms and their corresponding weights from the document collection index.

Building the index. To enable keystroke autocompletion, we use an n-gram index strategy. This strategy allows us to do matching against term prefixes. The maximal gram length is 20. As an example, let `protein` be a term that we want to index and assume that we use 2-grams. Then, our gram index would be: `_p`, `pr`, `ro`, `ot`, `te`, `ei`, `in`, in addition to `protein`.

Prefix and fuzzy queries. The use of n-gram indexes enables prefix queries and allows us to retrieve terms that share their prefixes with input sequences of characters. In addition to allowing higher fault tolerance, we also allow fuzzy queries. This feature allows us to retrieve terms that are similar to the typed one. The closeness is measured using the Levingston distance (also called edit distance).

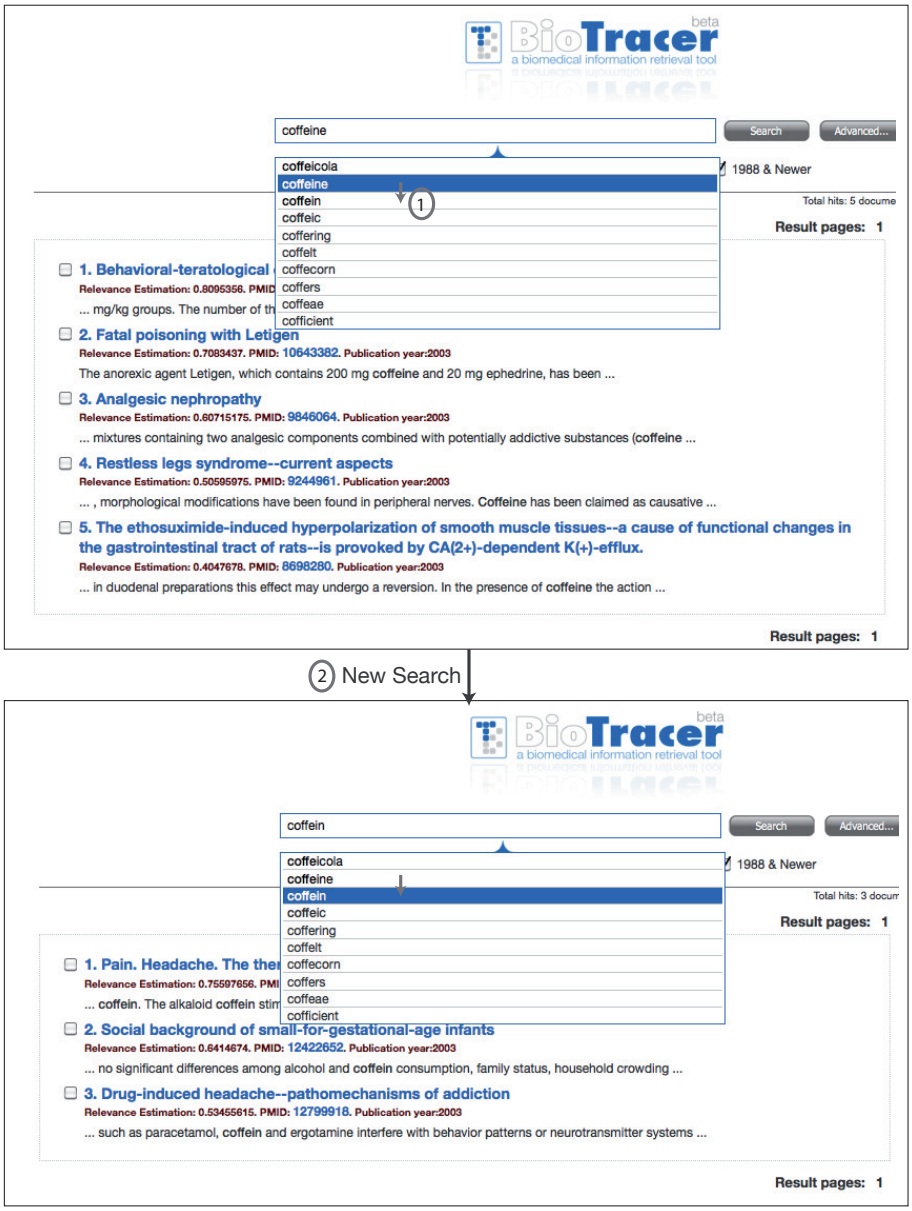

Fig. 2. Illustration of the autocompletion feature

Ranking. Once retrieved, the terms are ranked based on several factors. We mainly rank them based on their term weights stored in the main index. As a user types, words with most similar prefixes are readily retrieved first. Then, for terms that do not share prefixes, we use edit distance to order

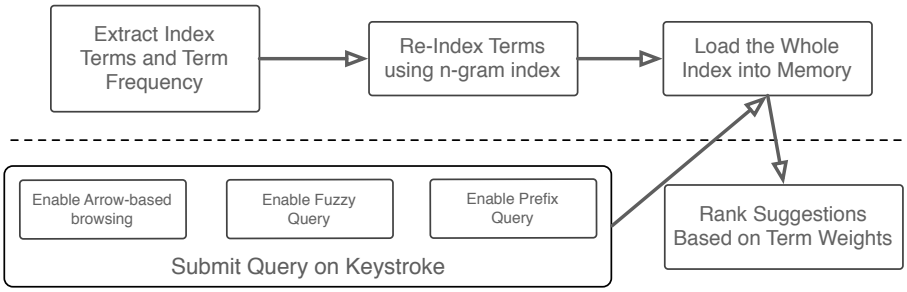

Fig. 3. Summary of the approach to achieve interactive autocompletion

terms that are related as specified by the fuzzy query. Term weights are the TF-IDF weight when we use the vector space model (VSM), while we use the weight in Eq. 4 for our BM25-based ranking model.

5 Evaluation

5.1 Prototype Implementation

In order to test our ideas, we implemented a proof-of-the-concept prototype, called BioTracer. Instead of re-inventing the wheel, we decided to use existing open source libraries to implement the prototype. Because of the requirements for performance, extensibility, simplicity, as well as scalability, we chose Java Lucene.

Figure 4 shows the BioTracer architecture. We use Lucene as a basis for indexing documents and handling queries, extended with a graphical user interface (GUI) based on JSP (Java Sever Pages) and AJAX (Asynchronous JavaScript and XML). The index is constructed mainly based on the MEDLINE database, which is parsed and handled by the Document Handler. To facilitate the parsing and generation of Lucene-friendly documents, we use Lingpipe[8] MEDLINE parser as a base for implementing the Document Handler. This handler also interacts with both the Language Model Handler and IndexManager to index documents based on a chosen language or similarity model.

All search is logged in the Search Log repository. This is used to "learn from experience", allowing BioTracer to use data from previous searches and user choices/interactions (e.g., URF). The Log is implemented using MySQL.

Figure 5 shows the implemented Web-based GUI, which also illustrates the URF possibility as described above. Each returned relevant hit can be marked by the user. The system will then use this information to re-rank the results.

Note that due to US National Library of Medicine (NLM) license regulations, BioTracer is not allowed to provide abstracts directly from MEDLINE. However, we have implemented a workaround that still allows users to browse the abstracts

[8] See http://alias-i.com/lingpipe/

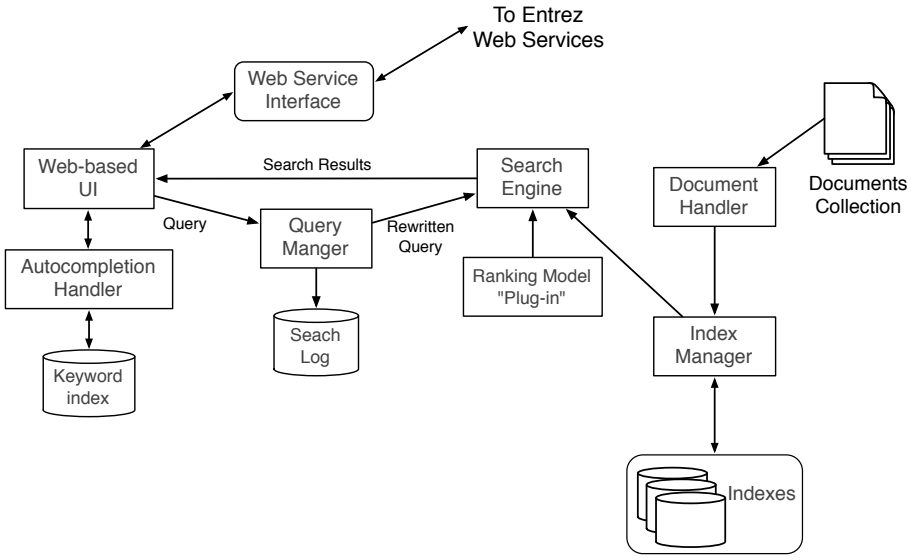

Fig. 4. The BioTracer architecture

Fig. 5. Screen dump showing the BioTracer search environment

inside BioTracer. The solution is using the Entrez Web Services interfaces in combination with AJAX. This allows us to retrieve a specific abstract from PubMed based on the PubMed ID (if available) and process it. In this way, we are able to allow users to study the abstracts inside BioTracer, rather than going to another Web page. Moreover, if the user wants to examine more than one abstract, our AJAX implementation allows several PubMed abstract windows to be popped up at a time.

5.2 Experimental Results

Experimental Setup: To test the retrieval performance of BioTracer, we have performed the evaluation based on the TREC 2004 Genomic Track [24] test collection. The main reason for choosing this collection is its number of topics covered, which span from gene name searches to disease searches. It allows us to do a comprehensive evaluation of the BioTracer retrieval performance.

The TREC 2004 collection originally consists of 4,591,008 MEDLINE documents. Our evaluation was based on a subset of 42,255 documents that were judged against 50 topics. Each topic consists of *title*, *need*, and *context* fields. The queries were generated manually from the topics using all the fields. The runs were carried out as follows. First, we tested the generated queries against an off-the-shelf Lucene similarity model. Then, we used the same queries on the extended models that we discussed earlier.

To evaluate the retrieval performance, we used the same approach implemented for Text REtrieval Conference (TREC), that is, by focusing on TREC mean average precision (MAP). For each query, we may get the average precision by computing the average precision value for the set of top-k documents after each document is retrieved. Thus, MAP is the mean of the average precision values for all the queries. MAP has been widely used as the measure to evaluate ad hoc retrieval results [25, 26]. It has also been used in the TREC Genomics Track evaluation [24]. Since MAP is sensitive to the rank of every relevant document, it reflects well the overall ranking accuracy. For this reason, the best result is the one with highest MAP values.

Further, since users are generally most interested in the results within the top 100 retrieved documents, we have stressed measuring the BioTracer's ability to find relevant data among the top 100 hits. Therefore, we measured specifically the precision at 10 retrieved documents (P@10) and precision at 100 (P@100).

To make our evaluation as independent as possible and to be able to compare it with previous results, we used the Buckley's **trec_eval** program[9]. Using trec_eval, the maximum number of allowed retrieved documents for each query was, by default, set to 1000. For each run, documents without any specific judgement were treated as non-relevant.

Results: Table 2 summarizes the results from our evaluation. The first column is the result from running the baseline method using the LuceneTFIDF-based

[9] See http://trec.nist.gov/trec_eval/

similarity model (VSM). The next column shows the results after we modified this model with the same extensions as those for BM25. Furthermore, the third column contains the result from running the baseline BM25 model. The result from our extensions in Sections 2, 3, and 3.4 are presented in the "Boolean+BM25" column, and the results from using user relevance feedback is shown in the "URF BM25" column. In addition to Table 2, Figure 6 shows the average precision values with respect to the number of retrieved documents and the used method. The different number of retrieved documents are represented by precision points. As an example, a precision point 20 means that we calculated the average precision at 20 retrieved documents.

Note that in addition to these results, we have tested the BioTracer prototype using Divergence From Randomness (DFR) BM25 [15], and Lemur TF-IDF models [14] with boolean and document boosts. However, the results from these runs were not significantly different from those of Boolean BM25. Therefore, they are not included in this paper.

Table 2. Results from running based on pooled TREC 2004 corpus

	Baseline Lucene TFIDF	Custom TFIDF	Baseline BM25	Boolean +BM25	URF BM25
MAP	0.346	0.4975	0.398	0.5122	**0.5129**
R-precision	0.3837	0.5443	0.4547	**0.558**	0.5527
P@10	0.534	0.652	0.594	0.666	**0.712**
P@100	0.3464	0.4552	0.3974	0.469	**0.4734**
Total recall	0.67	0.737	0.637	0.735	**0.738**

The results in Table 2 and Figure 6 show that there are only slight differences among the extended runs, although the URF BM25 had the overall best retrieval performance. These differences were most obvious for the top-15 hits. We can further see that the extended models performed much better than the baseline ones, which also show the effects of applying our extensions. For the TF-IDF model (i.e., the vector space model), the improvement of our custom TFIDF over the baseline model was 43.8%. The improvements for the BM25-based models were 28% and 29% respectively from the baseline model to the extended BM25 model and from the baseline model to URF BM25. Focusing on these results, adding the extensions is useful, in terms of retrieval performance. Further, comparing to the results from TREC 2004 Genomic Track [24] and assuming similar experiments, all of our extended models performed better than the best run from this track, where the MAP, P@10 and P@100 were to 0.4075, 0.604 and 0.4196, respectively [24].

Despite the above promising results, there are issues that we have to address. First, our evaluation was based on a somehow "closed" test collection. Although we have argued its comprehensiveness, we recognize that such a collection would hardly cover absolutely all aspects of users' needs in real search environments.

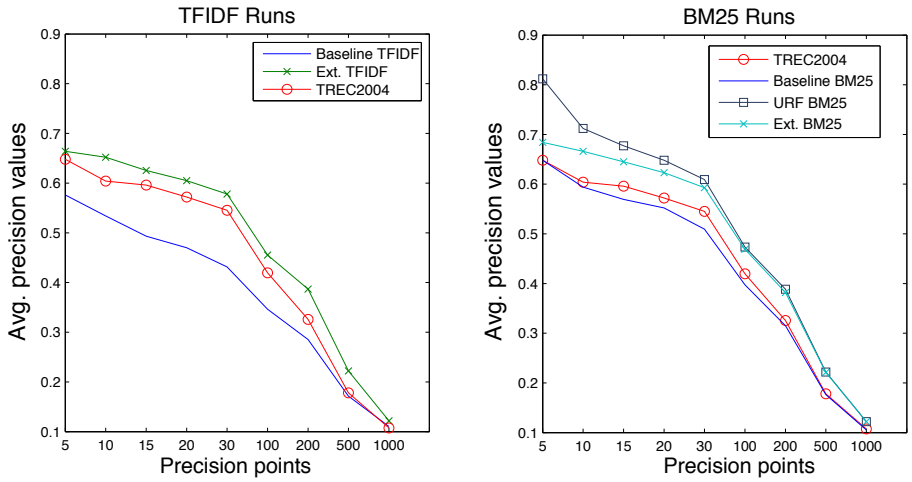

Fig. 6. Graphs for top-k precision values

This in combination with the fact that the BioTracer prototype can be seen as a highly interactive information search system, the generality/completeness and validity of the collection might be debatable. A way to make our evaluation more general is to involve real users, and study how they interact with BioTracer, as well as their evaluation of the retrieval performance. However, it is still not possible to guarantee a perfect evaluation [27], since we have to rely on several factors [28] such as the users' willingness to answer the interviews, the completeness of the questionnaire, and the broadness of the users.

Execution Speed: With respect to execution speed, although we argued the necessity of having an information retrieval system that can handle high-speed search (see Section 4.2), it has not been the main focus of this work. Rigorous performance tests, including comparison with other approaches, are needed to assess our system performance.

Still, our informal tests have shown promising results. We tried to run our tests using 4.5 Millions documents, a subset of MEDLINE entries on an Apple iMac machine with 2.93GHz Intel i7 CPU (quad core) and 8GB RAM. Using the aforementioned queries from the Pubmed one-day query log [23] (see Section 3.4), the search speeds were measured as follows. First, focusing on the speed as function of the document frequency, which varied from 1 document to 1.3 Millions documents and with one term in each query, the execution speed was constantly below 100 ms. Most specifically, the execution speed was 68 ms at maximum and 3.6 ms in average. Second, focusing on the speed as function of the number of query terms, our system managed to run our queries within 150 ms in average, when we varied our query lengths from 1 to 25 terms, whereas the maximum speed was 657 ms. Here, we got the lowest speeds when the number of queries were higher than 20. We believe the main reason for this is the way Lucene parses and process the queries.

In summary, the above results shows that our system is able to execute any query very fast independent of the number of returned results as long as the we keep the number terms in the query low (i.e., below 20 terms).

6 Related Work

There are many methods and systems suggested and developed in the biomedical information retrieval research domain. Most previous work has been presented in several TREC Genomic Track conferences, including the aforementioned TREC 2004 [24]. Other approaches applying tokenization heuristics were studied by Jiang and Zhai [10]. Their evaluation has shown that these methods can successfully improve the retrieval performance.

Like ours, many existing approaches have focused on expanding queries [24, 29]. However, we have attempted to go further by combining the use of different ranking models that can be adapted to different types of information needs, boosting different parts in the documents, applying a query-document correlation factor, using wildcard queries as query extension rather than stemming, using boolean queries with the BM25, and involving the users by user relevance feedback.

Concerning retrieval systems, there are several related systems. Because these systems mainly use proprietary ranking and indexing strategies, our discussion will focus on the system features and functionality.

First, perhaps the most used retrieval tool is PubMed[10]. PubMed provides several useful features for searching biomedical publications, such as the use of MeSH terms to expand queries and the possibility to access related articles. However, to our best knowledge, it does not provide ranking based on document relevance, per se. Instead, PubMed seems to use a proprietary algorithm to order the search results based on publication date, author names, and journals. Nevertheless, a more useful Web-based GUI extension to PubMed is available, called HubMed [30]. Like BioTracer, HubMed implements web service facilities for PubMed.

Another promising system is iPubmed [31]. The iPubmed system focuses on providing high performance and interactive search interface to MEDLINE data, through a "search-as-you-type" feature. It means that a search is initiated as soon as a keystroke is hit. Another useful feature of this system is their implementation of fuzzy search, allowing correct retrieval of documents even though users have misspelled their queries. Somehow, our autocompletion feature is inspired by this system with respect to instant and fuzzy search. The main difference is how the search results are retrieved and the ranking strategy applied.

Third, ScienceDirect is a search system[11]. It makes use of the powerful Scirus search facility. Scirus uses a more sophisticated ranking algorithm, which seems to produce more relevant results than PubMed. To our best knowledge, the main difference with BioTracer is that ScienceDirect/Scirus does not allow any

[10] See http://www.pubmed.org
[11] See http://www.sciencedirect.com

kind of user relevance feedback to refine the search. It is also worth noting that SienceDirect is a commercial system that covers a much larger scientific area than BioTracer. Therefore, the search results from this systems often include other than biomedical documents.

A fourth system worth discussing is Textpresso [32], which is also a search engine allowing specialized search of biomedical information. In contrast to BioTracer, however, Textpresso extensively uses ontologies in the retrieval process. For a biomedical text, each word or phrase is marked by a term in the ontology when they are indexed. Its ranking algorithm is based on the frequencies of queried index terms. This means that the document containing the largest number of query terms is ranked at the top. To our best knowledge, Textpresso does not offer any user relevance feedback feature.

Fifth, BioIE [33] is a rule-based information retrieval system that extracts informative sentences from the biomedical literature such as MEDLINE abstracts. It uses MEDLINE as the main underlying searchable information. In addition, it allows users to upload their own text to be searchable. Both statistical analysis (word distribution, filtered word distribution, N-gram distribution and MeSH term distribution) and sentence extraction can be performed. BioIE extracts informative sentences based on predefined templates for a user-specified category. The ranking algorithms of BioIE are based on the numbers of occurrences of terms in the query, much like Textpresso. BioIE does not support user relevance feedback.

7 Conclusion and Future Work

In this paper we presented our ongoing work towards the development of an information retrieval system, called BioTracer. It is a search prototype that allows customized ranking of search results through boosting specific parts of documents, customizing scoring schemes, and allowing users to affect the ranking through user relevance feedback (URF). In summary, we have investigated the effect of using and/or extending existing models like TF/IDF-based models and BM25 with support for boolean queries, boosting documents and document parts, as well as enabling URF on retrieval performance and result ranking. Focusing on URF and the ability to customize the retrieval process, users are more flexible to specify their views of relevance. Thus we can address the challenges that we face in searching biomedical information. In this respect, the major contribution of this work is the development of the BioTracer search system providing adaptable searching and ranking of biomedical information. Although we built the system on existing techniques, to our best knowledge, the way we integrated them and extended them in one system is unique. Moreover, we have done experiments with the TREC collection to investigate the system's retrieval performance. We believe that these experiments on biomedical information, leading to our comparison of the different ranking models and their retrieval characteristics, are in itself an interesting result. Nevertheless, focusing on BioTracer as a whole, the main conclusion from these experiments is that BioTracer is a tool

that is able to retrieve relevant information and that our extensions have helped improving the retrieval performance.

There are still challenges left for further studies. First, the system has only been tested against a TREC corpus. As discussed earlier, the strength and the validity of such a test can be improved. Therefore, we recognize the necessity of doing more empirical experiments with real users in a realistic environment. This additional work will reveal areas where our work can be improved. Further, we will further investigate the effect of including natural language processing (NLP) methods in the document handling process. This includes applying named-entity recognition (NER) to identify important keywords and sentences, thus further improving the way parts of a document are boosted. We are also investigating the use of controlled vocabularies such as MeSH [8] in both for query term boosting and extending queries. Finally, we will include a text classification facility. Our preliminary experiments based the Support Vector Machine (SVM) have, by far, shown good potential in terms of increased search precision. However, more work is needed to make such approaches efficient enough so that it does not degrade the overall system performance.

Acknowledgements. The author would like to thank Jon Olav Hauglid, Roger Midtstraum, Reidar Conradi, and Min-Yen Kan for their suggestions to improve this paper.

References

1. Ramampiaro, H.: BioMedical information retrieval: The bioTracer approach. In: Khuri, S., Lhotská, L., Pisanti, N. (eds.) ITBAM 2010. LNCS, vol. 6266, pp. 143–157. Springer, Heidelberg (2010)
2. Krauthammer, M., Nenadic, G.: Term identification in the biomedical literature. Journal of Biomedical Informatics 37(6), 512–526 (2004)
3. Chen, L., Liu, H., Friedman, C.: Gene name ambiguity of eukaryotic nomenclatures. Bioinformatics 21(2), 248–256 (2005)
4. Netzel, R., Perez-Iratxeta, C., Bork, P., Andrade, M.A.: The way we write. EMBO Reports 4(5), 446–451 (2003)
5. Baeza-Yates, R.A., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley Longman Publishing Co., Inc., Boston (1999)
6. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Information Processing and Management 24(5), 513–523 (1988)
7. Croft, B., Metzler, D., Strohman, T.: Search Engines: Information Retrieval in Practice, 1st edn. Addison-Wesley, Reading (2009)
8. Lowe, H.J., Barnett, G.O.: Understanding and using the medical subject headings (MeSH) vocabulary to perform literature searches. JAMA 271(14), 1103–1108 (1994)
9. Trieschnigg, D., Kraaij, W., de Jong, F.: The influence of basic tokenization on biomedical document retrieval. In: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2007, p. 803 (2007)
10. Jiang, J., Zhai, C.: An empirical study of tokenization strategies for biomedical information retrieval. Information Retrieval 10(4-5), 341–363 (2007)

11. Baeza-Yates, R.A., Ribeiro-Neto, B.: Modern Information Retrieval, 2nd edn. Addison-Wesley Longman Publishing Co., Inc., Boston (2011)
12. Hatcher, E., Gospodnetic, O.: Lucene in Action. Manning Publications Co., 209 Bruce Park Ave., Greenwich, CT 06830 (2005)
13. Robertson, S.E., Jones, K.S.: Simple proven approaches to text retrieval. Technical Report 356, University of Cambridge (1994)
14. Zhai, C.: Notes on the lemur TFIDF model. note with lemur 1.9 documentation. Technical report, School of CS, CMU (2001)
15. Amati, G., Rijsbergen, C.J.V.: Probabilistic models of information retrieval based on measuring the divergence from randomness. ACM Transactions on Information Systems 20(4), 357–389 (2002)
16. Ponte, J.M., Croft, W.B.: A language modeling approach to information retrieval. In: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 1998, pp. 275–281. ACM, New York (1998)
17. Wilkinson, R.: Effective retrieval of structured documents. In: Proceedings of the 17th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1994), pp. 311–317. Springer-Verlag New York, Inc., New York (1994)
18. Robertson, S., Zaragoza, H., Taylor, M.: Simple BM25 extension to multiple weighted fields. In: CIKM 2004: Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management, pp. 42–49. ACM, Washington, D.C., USA (2004)
19. Cohen, A.M., Hersh, W.R.: A survey of current work in biomedical text mining. Briefings in Bioinformatics 6(1), 57–71 (2005)
20. Leser, U., Hakenberg, J.: What makes a gene name? Named entity recognition in the biomedical literature. Briefings in Bioinformatics 6(4), 357 (2005)
21. Kabiljo, R., Clegg, A., Shepherd, A.: A realistic assessment of methods for extracting gene/protein interactions from free text. BMC Bioinformatics 10(1), 233 (2009)
22. Johannsson, D.V.: Biomedical information retrieval based on document-level term boosting. Master's thesis, Norwegian University of Science and Technology (NTNU) (2009)
23. Herskovic, J., Tanaka, L., Hersh, W., Bernstam, E.: A day in the life of PubMed: Analysis of a typical days query log. Journal of the American Medical Informatics Association 14(2), 212–220 (2007)
24. Hersh, W.R., Bhupatiraju, R.T., Ross, L., Roberts, P., Cohen, A.M., Kraemer, D.F.: Enhancing access to the bibliome: the trec 2004 genomics track. Journal of Biomedical Discovery and Collaboration 1(3), 10 (2006)
25. Yilmaz, E., Aslam, J.A.: Estimating average precision when judgments are incomplete. Knowledge and Information Systems 16(2), 173–211 (2008)
26. Voorhees, E.M.: On test collections for adaptive information retrieval. Inf. Process. Manage. 44(6), 1879–1885 (2008)
27. Käki, M., Aula, A.: Controlling the complexity in comparing search user interfaces via user studies. Information Processing and Management 44(1), 82–91 (2008); Evaluation of Interactive Information Retrieval Systems
28. Kelly, D., Harper, D.J., Landau, B.: Questionnaire mode effects in interactive information retrieval experiments. Information Processing and Management 44(1), 122–141 (2008); Evaluation of Interactive Information Retrieval Systems
29. Abdou, S., Savoy, J.: Searching in Medline: Query expansion and manual indexing evaluation. Information Processing & Management 44(2), 781–789 (2008)

30. Eaton, A.D.: Hubmed: a web-based biomedical literature search interface. Nucleic Acids Research 34(Web Server issue), W745–W747 (2006)
31. Wang, J., Cetindil, I., Ji, S., Li, C., Xie, X., Li, G., Feng, J.: Interactive and fuzzy search: a dynamic way to explore medline. Bioinformatics 26(18), 2321–2327 (2010)
32. Muller, H.M., Kenny, E.E., Sternberg, P.W.: Textpresso: an ontology-based information retrieval and extraction system for biological literature. PLoS Biol. 2(11), e309 (2004)
33. Divoli, A., Attwood, T.K.: BioIE: extracting informative sentences from the biomedical literature. Bioinformatics 21, 2138–2139 (2005)