Journal of Biomedical Informatics 46 (2013) 929–939

Contents lists available at ScienceDirect

Journal of Biomedical Informatics

journal homepage: www.elsevier.com/locate/yjbin

Development and evaluation of a biomedical search engine using a predicate-based vector space model

Myungjae Kwak [a,*], Gondy Leroy [b,d], Jesse D. Martinez [c], Jeffrey Harwell [b]

[a] School of Information Technology, Middle Georgia State College, Macon, GA 31206, United States
[b] School of Information Systems and Technology, Claremont Graduate University, Claremont, CA 91711, United States
[c] Cell Biology and Anatomy, Radiation Oncology, University of Arizona Cancer Center, Tucson, AZ 85719, United States
[d] Department of Management Information Systems, University of Arizona, Tucson, AZ 85721, United States

ARTICLE INFO

Article history:
Received 1 February 2013
Accepted 19 July 2013
Available online 25 July 2013

Keywords:
Search engine
Triple
Predicate
Information retrieval
Vector space model

ABSTRACT

Although biomedical information available in articles and patents is increasing exponentially, we continue to rely on the same information retrieval methods and use very few keywords to search millions of documents. We are developing a fundamentally different approach for finding much more precise and complete information with a single query using predicates instead of keywords for both query and document representation. Predicates are triples that are more complex datastructures than keywords and contain more structured information. To make optimal use of them, we developed a new predicate-based vector space model and query-document similarity function with adjusted *tf-idf* and boost function. Using a test bed of 107,367 PubMed abstracts, we evaluated the first essential function: retrieving information. Cancer researchers provided 20 realistic queries, for which the top 15 abstracts were retrieved using a predicate-based (new) and keyword-based (baseline) approach. Each abstract was evaluated, double-blind, by cancer researchers on a 0–5 point scale to calculate precision (0 versus higher) and relevance (0–5 score). Precision was significantly higher ($p < .001$) for the predicate-based (80%) than for the keyword-based (71%) approach. Relevance was almost doubled with the predicate-based approach—2.1 versus 1.6 without rank order adjustment ($p < .001$) and 1.34 versus 0.98 with rank order adjustment ($p < .001$) for predicate—versus keyword-based approach respectively. Predicates can support more precise searching than keywords, laying the foundation for rich and sophisticated information search.

1. Introduction

The availability of online medical and biomedical information has increased dramatically in recent years [1]. For example, PubMed currently contains over 19 million articles covering biomedicine and health-related research, and adds approximately 0.7 million abstracts per year. Although search engines have been developed into highly efficient and effective tools, the availability of more advanced underlying data structures and associated user interfaces would make paradigm shifting improvements and alternate uses possible.

Search engines and digital libraries are focused on, but also limited to, using strings of words. This is reflected in each user interface—users are limited to typing a list of words, and, at most, can indicate which words need to be combined or excluded by using quotes or “not.” This limitation is a consequence of the underlying phrase-based index which requires documents to be matched to words and phrases and it lends itself to a user interface which encourages suboptimal user search habits. For example, it has been shown that people continue to use very few keywords, only 2–3 on average [2–7], regardless of the topic of our search [7]. This existing keyword search technique results in imprecise queries as shown in Section 6.3.

To remedy the short queries, query expansion techniques have been researched and are currently used by search engines. Varying degrees of success are achieved when adding different numbers of keywords [8–10], using the most frequent terms [11], or using terms from different parts of documents [12–14]. However, people do not like automated methods [15] and so interactive query expansion is preferred. This has led to the modern query expansion, popularized by many search engines, to select a query from a popup with queries already used by others [16,17].

Unfortunately, today’s query expansion reduces the overall diversity of searches thereby also reducing the information available. Exacerbating the problem is that many search engines ignore portions of the queries. In most conventional keyword search engines, relationships between keywords are not captured or used to retrieve and rank the documents. Consequently, the search results show irrelevant results that contain the keywords but not

* Corresponding author.
E-mail addresses: myungjae.kwak@maconstate.edu (M. Kwak), gondy.leroy@cgu.edu, gondyleroy@email.arizona.edu (G. Leroy), jmartinez@azcc.arizona.edu (J.D. Martinez), jeffrey.harwell@cgu.edu (J. Harwell).

1532-0464/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jbi.2013.07.006

the relationship between them. For example, using PubMed to find articles about "*NBS1 interacts with endocytic proteins to affect the central nervous system*," the query "*NBS1, endocytosis, central nervous system,*" results in 329 documents of which only one of the top 20 documents were related to the query intent. In this example, the relation "*interacts with*" between "*NBS1*" and "*endocytic protein,*" and the relation "*affect*" between "*NBS1*" and "*central nervous system*" were not captured, and as a result most of the retrieved documents are about irrelevant matters (DNA damage, functional polymorphisms in the NBS1, etc.).

Many improvements are possible, such as recognizing entities or visualizing results, and they are the topic of much research and development. Our work focuses on one such aspect—the inclusion of the relationship between the keywords in a search query. We define such a relationship as a predicate and store these as triples in the search engine index. The triples are a natural way of describing the vast majority of online data and resources [18]. Moreover, triples consisting of subject, verb or preposition (predicate), and object and form an elementary sentence that represents the basic information of the search [1,18].

This study describes the development and evaluation of a predicate-based search engine that uses predicates in addition to keywords. Evaluated using a collection of more than 100,000 Medline abstracts the results showed that a basic implementation of this new, hybrid approach outperformed a basic implementation of the baseline approach (keyword-based search). In our study, the average precision of the predicate-based search was significantly improved by 9.17% compared to that of a keyword-based approach. The average relevance of the predicate-based search was significantly improved by 31.25% compared to that of a keyword-based approach. Our pilot study [19] showed for three examples that our approach to predicate-based search is an improvement. The contribution of this work is the further improvement of algorithms which are then evaluated with a new user study demonstrating the strength of predicate-based searching in biomedicine, a necessary first step in laying a foundation for future general search mechanisms. Although the approach can be applied to other types of text, scientific texts that describe relationships between variables are highly suited to our project.

2. Related work

2.1. Search engine component overview

The search engine field is diverse with applications ranging from finding a document to finding a new house or partner. While many different text search engines and digital libraries exist, all of their main components are the same—they match input items, usually words, to elements stored in their collection. Text search engines store documents, or links to those documents, and indexes. User queries are interpreted and related to the documents by means of those indexes [20,21]. Most search engines operate on vast collections of documents and various indexing and searching algorithms, such as the Boolean retrieval model, term-document incidence matrix, and inverted index, are required to find the desired documents from the collection with sufficient accuracy and speed. For example, in the Boolean retrieval model, queries are represented in the form of a Boolean expression of terms using operators such as AND, OR, and NOT, and the search results are derived by posing the queries against the term-document incidence matrix.

Indexing is used to avoid linearly scanning all available documents for each query. The term-document incidence matrix, which consists of documents, terms, and frequencies of the terms in each document, is the most popular way of indexing [20,21]. The size of this incidence matrix increases enormously as more documents are added to the collection. However, since the data in this matrix is sparse, an inverted index is used to record only the non-zero entries of the matrix. The major components in the inverted index are the dictionary, i.e., the list of terms, term frequency, and the link from the term to the original documents, i.e., the identifiers' list of documents that contains the term. Search engines use this index to find documents and rank them giving a higher weight to the documents with the highest frequency of the search terms [20]. This inverted index is used to represent term weights in the vector space model and is the most commonly implemented model of most current search engines [21].

Meanwhile, many approaches have been suggested and discussed to address the word mismatch problem of the traditional word matching methods. Query expansion techniques improve search performance by evaluating user search queries and expanding them with additional terms [22]. These techniques also include the use of synonyms or additional morphological forms of terms in the query [23,24]. Other approaches have focused on linguistic concepts, which can be defined in lexical resources such as WordNet [25]. Latent Semantic Indexing (LSI), for example, aims to extract latent concepts from the text, construct meaningful groups of words, and search for such latent concepts by applying Singular Value Decomposition (SVD) to the original term-document matrix [26]. Latent Dirichlet Allocation (LDA) is a probabilistic latent topic model whose basic idea is to represent multi-lingual documents by a mixture of latent concepts or topics [24,27–29]. While query expansion or approaches based on latent concepts have become common, these techniques rely on reusing queries from others or lexical resources based on concepts and topics, not additional user information.

2.2. Vector space model

While the above improvements are possible for search engines, our work focuses on the central component—the representation of information in the underlying index and how to match this to the user's query. We first review the current approach—the vector space model. To calculate the similarity between query and document search engines use the vector space model which represents both query and document as a vector of keywords. Weights represent the importance of the keywords in document d and query q within the entire document collection [30].

$$d_i = (w_{i1}, w_{i2}, \ldots, w_{it}) \tag{1}$$

$$q = (w_{q1}, w_{q2}, \ldots, w_{qs}) \tag{2}$$

Term weights can be defined in various ways in a document vector. The common, basic approach is to use the *tf-idf* method [31] in which the weight of a term is determined by two factors, (1) how often the term j occurs in a document d_i (term frequency $tf_{i,j}$) combined with (2) how often the term j occurs in the document collection (document frequency df_j). Document frequency df_j is needed to scale a term's weight in the document collection. Denoting the total number of documents in the document collection by N, the inverse document frequency (idf_j) of a term j can be defined as in (3). This makes the *idf* of a rare term high but makes the *idf* of a frequent term low [20]. The composite weight for a term in each document combines term frequency and inverse document frequency. Thus, when using *tf-idf* weighting, the weight of a term j in document d_i is defined by (4).

$$idf_j = \log \frac{N}{df_j} \tag{3}$$

$$w_{i,j} = tf_{i,j} \times idf_j = tf_{i,j} \times \log N/df_j \tag{4}$$