Set-Based Vector Model: An Efficient Approach for Correlation-Based Ranking

BRUNO PÔSSAS, NIVIO ZIVIANI, and WAGNER MEIRA, JR.
Federal University of Minas Gerais, Brazil
and
BERTHIER RIBEIRO-NETO
Federal University of Minas Gerais, Brazil and Akwan Information Technologies

This work presents a new approach for ranking documents in the vector space model. The novelty lies in two fronts. First, patterns of term co-occurrence are taken into account and are processed efficiently. Second, term weights are generated using a data mining technique called association rules. This leads to a new ranking mechanism called the *set-based vector model*. The components of our model are no longer index terms but index termsets, where a termset is a set of index terms. Termsets capture the intuition that semantically related terms appear close to each other in a document. They can be efficiently obtained by limiting the computation to small passages of text. Once termsets have been computed, the ranking is calculated as a function of the termset frequency in the document and its scarcity in the document collection. Experimental results show that the set-based vector model improves average precision for all collections and query types evaluated, while keeping computational costs small. For the 2-gigabyte TREC-8 collection, the set-based vector model leads to a gain in average precision figures of 14.7% and 16.4% for disjunctive and conjunctive queries, respectively, with respect to the standard vector space model. These gains increase to 24.9% and 30.0%, respectively, when proximity information is taken into account. Query processing times are larger but, on average, still comparable to those obtained with the standard vector model (increases in processing time varied from 30% to 300%). Our results suggest that the set-based vector model provides a correlation-based ranking formula that is effective with general collections and computationally practical.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications—*Data mining*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Retrieval models*

General Terms: Theory, Experimentation

This work was supported in part by GERINDO project–grant MCT/CNPq/CT-INFO 552.087/02-5, by CNPq scholarship 141.269/02-2 (Bruno Pôssas), by CNPq grant 520.916/94-8 (Nivio Ziviani), by CNPq grant 30.0188/95-1 (Berthier Ribeiro-Neto), and CNPq grant 30.9379/03-2 (Wagner Meira, Jr.).

Authors' addresses: B. Pôssas, N. Ziviani, and W. Meira, Jr., Department of Computer Science, Federal University of Minas Gerais, 31270-901, Belo Horizonte, MG, Brazil; B. Ribeiro-Neto, Akwan Information Technologies, Av. Antônio Abrahão Caram, 430 – 4o. andar, São José, 31.275-000, Belo Horizonte, MG, Brazil; email: {bavep,nivio,meira,berthier}@dcc.ufmg.hr.

Additional Key Words and Phrases: Information retrieval models, association rule mining, weighting index term co-occurrences, data mining, correlation-based ranking

1. INTRODUCTION

In the area of Information Retrieval (IR), the most popular models for ranking the documents of a collection (not necessarily a Web collection) are (i) the vector space models [Salton and Lesk 1968; Salton 1971], (ii) the probabilistic models [Maron and Kuhns 1960; van Rijsbergen 1979; Robertson and Jones 1976; Robertson and Walker 1994], and (iii) the statistical language models [Ponte and Croft 1998; Berger and Lafferty 1999; Lafferty and Zhai 2001]. The differences between these models rely on the representation of queries and documents, on the schemes for term weighting, and on the formula for computing a ranking.

Designing effective schemes for term weighting is a critical step in a search system if improved ranking is to be obtained. However, finding good term weights is an ongoing challenge. In this work we propose a new term weighting schema that leads to improved ranking and is efficient enough to be practical.

The best-known term-weighting schemes use weights that are a function of the number of times the index term occurs in a document and the number of documents in which the index term occurs. Such term-weighting strategies are called $tf \times idf$ (term frequency times inverse document frequency) schemes [Salton and McGill 1983; Baeza-Yates and Ribeiro-Neto 1999; Witten et al. 1999]. A modern variation of these strategies is the BM25 weighting scheme used by the Okapi system [Robertson and Walker 1994; Robertson et al. 1995].

All practical term-weighting schemes, to this date, assume that the terms are mutually independent—an assumption often made for mathematical convenience and simplicity of implementation. However, it is generally accepted that exploitation of the correlation among index terms in a document might be used to improve retrieval effectiveness with general collections. In fact, distinct approaches that take term co-occurrences into account have been proposed over time [Wong et al. 1985, 1987; van Rijsbergen 1977; Harper and van Rijsbergen 1978; Raghavan and Yu 1979]. All these approaches suffer from a common drawback, they are too inefficient computationally to be of value in practice.

In this article, we propose a new model for computing index term weights that takes into account patterns of term co-occurrence and is efficient enough to be of practical value. Our model is referred to as *set-based vector model*. For simplicity, we also refer to it as set-based model. Its components are termsets, where a termset is a set of the index terms of a collection of documents. To compute termsets, we propose an algorithm based on association rules theory [Agrawal et al. 1993]. Association rules are interesting because they provide all the elements of the $tf \times idf$ scheme in an algorithmically efficient and parameterized way. Moreover, they naturally provide for the quantification of representative patterns of term co-occurrence, something that is not present in

the $tf \times idf$ scheme. Some of the results of this paper were presented in [Pôssas et al. 2002a, 2002b, 2004].

The set-based model is the first information retrieval model that exploits term correlations effectively, provides significant gains in precision, and has processing costs close to the costs of the vector space model, independently of the collection and query type considered. The model exploits the intuition that semantically related terms often occur close to each other by implementing a pruning strategy that restricts computation to proximate termsets.

We validate the set-based model through experiments with two test collections: TREC-8 [Voorhees and Harman 1999] and WBR99. The TREC-8 collection is 2 gigabytes large and is composed of approximately 530,000 documents. The WBR99 collection is 16 gigabytes large and is composed of approximately 6 million pages from the Brazilian Web. We compare the standard vector space model to the set-based model.

For the TREC-8 collection, the set-based model yields gains in average precision of 14.7%, 16.4%, and 17.5% relative to the standard vector space model for processing disjunctive, conjunctive, and phrase queries, respectively. When only the top 10 documents of the answer set are considered, these gains go up to 28.2%, 27.6%, and 18.9%, respectively. The gains in average precision increase to 24.9% for disjunctive queries and to 30.0% for conjunctive queries, when proximity information is taken into account. If only the top-10 ranked documents are considered, these gains increase further to 35.29% and 30.80%, respectively.

For the WBR99 collection, the set-based model yields gains in average precision of 2.4%, 7.3%, and 8.9% relative to the standard vector space model for processing disjunctive, conjunctive, and phrase queries, respectively. When proximity information is considered the gains in average precision increase to 10.4% for processing disjunctive queries and to 11.0% for processing conjunctive queries. We notice that the gains in average precision are still consistent but are now smaller. The key reason is that Web queries are frequently too short, which limits the occurrence of termsets within the user query.

The set-based model is also competitive in terms of processing time. For the TREC-8 collection the increase in the average response time over the vector space model is approximately 37.2%, 33.2%, and 22.5% for processing disjunctive, conjunctive, and phrase queries, respectively. For the WBR99 collection, the increase in the average response time is just 19.3%, 21.0%, and 18.1%, respectively. The use of proximity information further increases processing times for the set-based model, because the positional index must be scanned. Processing times increase by 318.8% for disjunctive queries and by 312.8% for conjunctive queries, in the case of the TREC-8 test collection. For the WBR99 test collection, these increases go up to 431.4% and to 186.7%, respectively.

The remaining of the article is organized as follows. In the following section, we present related work. Section 3 discusses our method for computing termsets, which is based on association rules. In Section 4, the basic features of the set-based model are developed and justified. In Section 5, we show how different query types, such as conjunctive, disjunctive, and phrase queries, are processed in our model. Section 6 describes the reference collections we used.

The tuning of the set-based model is described in Section 7. Experimental results are presented in Sections 8 and 9. Finally, in Section 10, we draw our conclusions and suggestions for future research.

2. RELATED WORK

Different approaches to account for co-occurrence among index terms have been proposed. Raghavan and Yu [1979] use statistical analysis of a set of queries (considering relevant and nonrelevant document sets) to establish positive and negative correlations among index terms. The work by van Rijsbergen [1977] introduces a probabilistic model that incorporates dependences among index terms. Experimental results were later presented in a companion paper [Harper and van Rijsbergen 1978]. The extent to which two index terms depend on one another is derived from the distribution of co-occurrences in the whole collection, in the relevant document set, and in the nonrelevant document set, leading to a nonlinear weighting function. As shown in Salton et al. [1982], the resulting formula for computing the dependency factors in van Rijsbergen [1977] and Harper and van Rijsbergen [1978] does not seem to be computationally feasible, even for a relatively small number of index terms. The work in Bollmann-Sdorra and Raghavan [1998] presents a study of term dependence in a query space.

The work in Wong et al. [1985, 1987] presents an interesting approach to compute index term correlations based on automatic indexing schemes. It defines a new information retrieval model called *generalized vector space model*. The work shows that index term vectors can be explicitly represented in a 2^t-dimensional vector space, where t is the vocabulary size, such that index term correlations can be incorporated into the vector space model in a straightforward manner. It represents index term vectors using a basis of orthogonal vectors called min-terms, where each min-term represents one of the 2^t possible patterns of index term co-occurrence inside documents. Each index term k_i is represented by a term composed of all min-terms related to k_i. The model is not computationally feasible for moderately large collections because there are 2^t possible min-terms. Extensions of the generalized vector space model were presented in Alsaffar et al. [2000] and Kim et al. [2000].

Billhardt et al. [2002] present a context vector model that uses term dependencies in the process of indexing documents and queries. The context vectors that represent the documents of a collection provide richer descriptions of their basic characteristics. The similarity calculation is based on a semantic-matching rather than on a simple word-matching approach.

Language modeling approaches to information retrieval usually do not capture correlation between terms. However, there have been attempts to represent correlation among index terms using *bigrams* or *bi-terms* [Song and Croft 1999; Srikanth and Srihari 2002]. In these latter works, only adjacent words are assumed to be related. Nallapati and Allan [2002] and Gao et al. [2004] present alternative language models that allow representing term correlations. These correlations are bounded by a document sentence, such that only the strongest word dependencies are considered in order to reduce estimation errors.

Our work differs from the aforementioned works in the following aspects. First, determination of index term weights is derived directly from association rules theory, which naturally considers representative patterns of term co-occurrence (see Section 3). Second, in the set-based model term correlations were not restricted to adjacent, or sentence bounded words, all valid/genuine correlations between the query terms are taken into consideration. Third, experimental results (see Section 8) show significant and consistent improvements in average precision curves in comparison to the vector space model and to the generalized vector space model. These improvements in average precision do not always occur in the generalized vector space model because exhaustive application of term co-occurrences to all documents in the collection might eventually hurt the overall effectiveness and performance. Fourth, the set-based model algorithm is practical and efficient for queries containing up to 30 terms, with processing times comparable to the times to process the standard vector space model (see Section 9).

The set-based model is the first information retrieval model that exploits term correlations and term proximity effectively and provides significant gains in terms of precision, regardless of the size of the collection and of the size of the vocabulary. All known approaches that account for correlation among index terms were initially designed for processing only disjunctive queries. The set-based model provides a simple, effective, efficient and parameterized way to process disjunctive, conjunctive, and phrase queries.

The work in Bollmann-Sdorra et al. [2001] introduces a theoretical framework based on Boolean retrieval for the mining of association rules. In here, we use association rules to define a variant of the vector space model that allows quantifying term co-occurrences and successfully processing disjunctive, conjunctive, and phrase queries in a natural fashion.

3. MODELING CORRELATION AMONG TERMS

In this section, we introduce the concept of termsets as a basis for modeling dependences among index terms in the set-based model. We also present two special types of termsets: proximate and closed termsets.

3.1 Termsets

Definition 3.1. Let $T = \{k_1, k_2, \ldots, k_t\}$ be the vocabulary of a collection C of documents, that is, the set of t unique terms that appear in all documents in C. There is a total ordering among the vocabulary terms, which is based on the lexicographical order of terms, so that $k_i < k_{i+1}$, for $1 \leq i \leq t - 1$.

Definition 3.2. An n-termset $S, S \subseteq T$, is a set of n terms. When the number of terms is not important, we refer simply to the termset S.

Definition 3.3. Let $V = \{S_1, S_2, \ldots, S_{2^t}\}$ be the vocabulary-set of a collection C of documents, that is, the set of 2^t unique termsets that may appear in any document from C. The frequency dS_i of a termset S_i is the number of occurrences of S_i in C, that is, the number of documents where $S_i \subseteq d_j$ and $d_j \in C, 1 \leq j \leq N$.

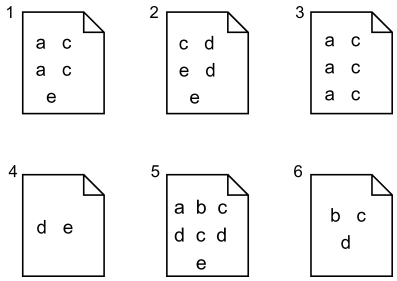


Fig. 1. Sample document collection.

Table I. Vocabulary-Set for the Query $q = \{a, b, c, d\}$

Termsets	Elements	Documents
S_a	$\{a\}$	$\{d_1, d_3, d_5\}$
S_b	$\{b\}$	$\{d_5, d_6\}$
S_c	$\{c\}$	$\{d_1, d_2, d_3, d_5, d_6\}$
S_d	$\{d\}$	$\{d_2, d_4, d_5, d_6\}$
S_{ab}	$\{a, b\}$	$\{d_5\}$
S_{ac}	$\{a, c\}$	$\{d_1, d_3, d_5\}$
S_{ad}	$\{a, d\}$	$\{d_5\}$
S_{bc}	$\{b, c\}$	$\{d_5, d_6\}$
S_{bd}	$\{b, d\}$	$\{d_5, d_6\}$
S_{cd}	$\{c, d\}$	$\{d_2, d_5, d_6\}$
S_{abc}	$\{a, b, c\}$	$\{d_5\}$
S_{abd}	$\{a, b, d\}$	$\{d_5\}$
S_{acd}	$\{a, c, d\}$	$\{d_5\}$
S_{bcd}	$\{b, c, d\}$	$\{d_5, d_6\}$
S_{abcd}	$\{a, b, c, d\}$	$\{d_5\}$

Definition 3.4. With each termset S_i, $1 \leq i \leq 2^t$, we associate an inverted list lS_i composed of the identifiers of the documents containing that termset. The frequency dS_i of a termset S_i can be computed as the length of its associated inverted list ($\mid lS_i \mid$). Further, we also use lS_i to refer to the set of documents in the list.

Definition 3.5. A termset S_i is a frequent termset if its frequency dS_i is greater than or equal to a given threshold, known as *support* in the scope of association rules [Agrawal et al. 1993] and referred to as minimal frequency in this work. As presented in the original Apriori algorithm [Agrawal and Srikant 1994], an n-termset is frequent if and only if all of its $(n-1)$-termsets are also frequent.

Example 3.1. Consider a vocabulary of five terms $T = \{a, b, c, d, e\}$, and a collection C of six documents d_j, $1 \leq j \leq 6$, given by $C = \{(a, c, a, c, e), (c, d, e, d, e), (a, c, a, c, a, c), (d, e), (a, b, c, d, c, d, e), (b, c, d)\}$, as depicted in Figure 1. Consider also the user query $q = \{a, b, c, d\}$. There are 15 termsets associated with q and all of them occur in our sample collection, as depicted in Table I. The 1-termsets are S_a, S_b, S_c and S_d, the 2-termsets are S_{ab}, S_{ac},

S_{ad}, S_{bc}, S_{bd}, and S_{cd}, the 3-termsets are S_{abc}, S_{abd}, S_{acd}, and S_{bcd}, and the only 4-termset is S_{abcd}.

A key feature of our approach is that we only compute the set of termsets associated with the query terms. The generalized vector space model [Wong et al. 1985, 1987], on the other hand, requires the computation of weights for all subsets of correlated terms in the document space, which is hard to compute with large collections. Thus, the set-based model computation is simpler and faster.

3.1.1 *Generating Termsets.* Our procedure for generating termsets is an adaptation of a well-known algorithm for determining termsets [Agrawal and Srikant 1994]. As mentioned, the main challenge in determining the frequent termsets is that the number of termsets increases exponentially with the number of distinct terms of the query, making naive or exhaustive approaches infeasible.

To search for frequent termsets, we use a simple and powerful principle: for an n-termset to be frequent, all $(n-1)$-termsets that are subsets of it must be frequent. Several of the most efficient data mining algorithms for association rules are based on this principle. They start by verifying which single terms are frequent and then combine them into 2-termsets. With each 2-termset is associated an inverted list of documents which is used to determine whether the 2-termset is frequent or not. The process iterates for termsets of size 3 and up, until there are no more frequent termsets to be found.

To determine whether a termset is frequent or not, a three-steps procedure is executed: (i) verify whether its subsets are frequent; if so, (ii) generate its inverted list and count its size; and (iii) check whether its inverted list size is above the minimum frequency. As expected, the most expensive task is the second one, which is implemented as an intersection of the inverted lists of the $(n-1)$-termsets that are subsets of the termset being generated.

Example 3.2. Consider our example document collection in Figure 1 and a minimum frequency equal to 2. To determine whether S_{bcd} is frequent, we first check whether S_{bc}, S_{bd}, and S_{cd} are frequent. Since they are indeed frequent, we generate lS_{bcd} by intersecting the lists for S_{bc}, S_{bd}, and S_{cd}. The resulting list contains the documents $\{d_5, d_6\}$. We can conclude that S_{bcd} is frequent, because its frequency is greater than or equal to the minimum frequency.

3.2 Proximate Termsets

We extend the concept of termsets to consider the proximity among the terms in the documents, as a strategy for generating termsets that are more meaningful. To store information on proximity among terms in a document, we extend the structure of the inverted lists as follows: For each term-document pair $[i, d_j]$, we add a list of occurrence locations of the term i in the document d_j, represented by $rp_{i,j}$, where the location of a term i is equal to the number of terms that precede i in document j. Thus, each entry in the inverted list for term i becomes a triple $<d_j, tf_{i,j}, rp_{i,j}>$.

To compute proximate termsets, we modify the algorithm for computing termsets by adding a new constraint: two terms are considered close when their distance is bounded by a proximity threshold, called minimum proximity. This technique is equivalent to the concept of intra-document passages [Zobel et al. 1995; Kaszkeil and Zobel 1997; Kaszkeil et al. 1999].

Proximity information works as a pruning strategy that limits termsets to those formed by proximate terms. This captures the notion that semantically related terms often occur close to each other. Verifying the proximity constraint is quite straightforward and consists of rejecting the termsets that contain terms whose distance is larger than the given threshold.

Example 3.3. To illustrate how proximity affects the determination of termsets, consider the termsets S_a and S_c in Example 3.1 and a minimum proximity threshold of 1. To verify whether S_{ac} is frequent, it is necessary to consider the proximity of the occurrences of a and c. Terms a and c co-occur in documents d_1, d_3, and d_5. We then calculate $rp_{a,1} = \{1, 3\}$, $rp_{c,1} = \{2, 4\}$, $rp_{a,3} = \{1, 3, 5\}$, $rp_{c,3} = \{2, 4, 6\}$, $rp_{a,5} = \{1\}$, and $rp_{c,5} = \{3, 5\}$. Following, we verify for each document whether the occurrences of the termsets S_a and S_c are within the proximity threshold. This is the case for documents 1 and 3, but not for document 5. Thus, the support of S_{ac} is set to 2. Clearly, the application of this new criterion tends to reduce the total number of termsets. Most important, the termsets that are computed represent stronger correlations, which tends to improve the retrieval effectiveness. Our experimental results (see Section 8) confirm such observations.

3.3 Termset Rules

Each termset embeds semantic information on term correlations. However, since some correlations may subsume others, the use of all of them may introduce noise into the model and reduce the retrieval precision. Termsets frequently overlap because a frequent termset implies that its subsets are also frequent. In Example 3.1, termset S_{ac} is a subset of termset S_{abc}, both are frequent, and there is an overlap in the set of documents in which they occur, more specifically $\{d_5\}$. One issue in this case is whether both termsets should be considered for retrieving information, since discarding one of them may result in information loss. We distinguish two scenarios where information loss may occur. First, if we discard S_{abc}, we lose information on the correlation among the terms a, b, and c. Second, if we discard S_{ac}, we also lose information on a correlation that is "popular" (it occurs in three documents) and thus, more meaningful for retrieval purposes.

In summary, whenever two termsets overlap and one termset is a subset of the other, discarding the larger termset results in losing correlation information. Discarding the smaller termset results in losing popularity information. To better understand this information loss process, we introduce the use of "rules", which are good for identifying precedence relations.

Definition 3.6. In the context of termsets, a rule is an implication $X \rightarrow Y$, where X and Y are termsets. The rule is characterized by a confidence

degree, which is the probability that Y appears in a document given that X has appeared.

Example 3.4. To illustrate, consider the rules $S_{ac} \rightarrow S_{abc}$ that has 33% confidence in our example collection of Figure 1 and $S_{bc} \rightarrow S_{bcd}$ that has 100% confidence. Discarding either of the termsets that compose the first rule will result in information loss. However, discarding S_{bc} in the second rule, while keeping S_{bcd}, will not result in any loss because the information carried by S_{bcd} is exactly the same information carried by S_{bc}. This discarding strategy reduces the number of termsets to be considered while yielding better retrieval results.

Whenever a termset rule has 100% confidence, the "smaller" termset may be discarded without information loss. This can be accomplished by enumerating all termset rules and then selecting those with 100% confidence. But, since enumerating all termset rules is expensive, it is necessary to devise a strategy for selecting termsets to be discarded. Closed termsets, an extension of the concept of frequent termsets, hold properties that allow implementing faster strategies for pruning out redundant termsets.

3.4 Closed Termsets

In this section we introduce the idea of closed termsets, an extension to the concept of frequent termsets.

Definition 3.7. The closure of a termset S_i is the set of all frequent termsets that co-occur with S_i in the same set of documents and that preserve the proximity constraint.

Definition 3.8. The closed termset CS_i is the largest termset in the closure of a termset S_i. More formally, given a set $\mathcal{D} \subseteq C$ of documents and a set $\mathcal{S}_\mathcal{D}$ of termsets that occur in all documents from $\mathcal{D}$ and only in these, a closed termset CS_i satisfies the property that $\nexists S_j \in \mathcal{S}_\mathcal{D} | (CS_i \subset S_j \wedge lS_i \equiv lS_j)$.

Closed termsets allow one to automatically discard termsets that do not aggregate any additional information of value. In fact, closed termsets encapsulate termsets that are the consequent in 100%-confidence rules. Closed termsets are interesting because they represent a reduction in the computational complexity and in the amount of data that has to be analyzed for ranking purposes, without loss of information.

Example 3.5. Consider the dataset of Example 3.1. Table II shows all frequent and closed termsets and their respective frequencies. If we define that a frequent termset must have a minimum frequency of 50%, the number of termsets is reduced from 15 to 6. Notice that the number of frequent termsets, although potentially very large, is usually small in natural language texts. Regarding the closed termsets, even in this small example, we see that the number of closed termsets is considerably smaller than the number of frequent termsets, for a minimum frequency of 17%.

A major advantage of using closed termsets, instead of frequent termsets, is that they can be generated very efficiently. Since the number of closed termsets

Table II. Frequent and Closed Termsets for the Sample Document Collection of Example 3.1

Frequency (ds)	Frequent Termsets	Closed Termsets
83% (5)	S_c	S_c
67% (4)	S_d	S_d
50% (3)	S_a, S_{ac}	S_{ac}
50% (3)	S_{cd}	S_{cd}
33% (2)	$S_b, S_{bc}, S_{bd}, S_{bcd}$	S_{bcd}
17% (1)	$S_{ab}, S_{ad}, S_{abc}, S_{abd}, S_{acd}, S_{abcd}$	S_{abcd}

is at most equal to the number of frequent termsets, we may also use this bound as an upper limit to the number of closed termsets. As discussed in Section 9, in practical situations, the number of closed termsets is significantly smaller than the number of frequent termsets.

3.4.1 *Generating Closed Termsets.* Determining closed termsets is an extension of the problem of mining frequent termsets. Our approach is based on an efficient algorithm called CHARM [Zaki 2000]. We adapt that algorithm to handle terms and documents instead of items and transactions, respectively.

The starting point of the algorithm is the set of frequent termsets for a document collection. Following a total ordering criterion, the lexicographic order in our case, we determine all possible closures, testing whether each termset is closed or not. Whenever a termset is subsumed by other within a closure, that termset is removed from the set of closed termsets.

Formally, assume two termsets S_i and S_j, where $S_i \leq S_j$ under our total ordering criterion. The comparison between lS_i and lS_j, the lists of documents associated with S_i and S_j, respectively, leads to one of the following two situations:

(1) if $lS_i = lS_j$, we verify whether the termset S_j is a subset of S_i. If it is, we can discard S_j without information loss. If it is not, we can remove S_i and S_j, replacing them by $S_{i\cup j}$ in the set of current closed termsets, since the closure of S_i and S_j is equal to the closure of $S_{i\cup j}$.
(2) if $lS_i \neq lS_j$, we cannot remove or discard anything, because the two termsets lead to different closures. There is only one possible action, that is to add S_i and S_j to the set of closed termsets.

Example 3.6. Consider our sample collection of Example 3.1. A lexicographic total ordering of the frequent termsets would be $S_a < S_{ab} < S_{abc} < S_{abcd} < S_{abd} < S_{ac} < S_{acd} < S_{ad} < S_b < S_{bc} < S_{bcd} < S_{bd} < S_c < S_{cd} < S_d$. The starting point of the algorithm is the set of frequent termsets. The set of closed termsets, denoted $\mathcal{C}$, is initially set to the empty set. To determine the closed termsets, we start by comparing S_a with the termset S_{ab} that comes after it. Since $lS_a \neq lS_{ab}$, both termsets are added to $\mathcal{C}$. Following, we compare S_{ab} with S_{abc}. Since $lS_{ab} = lS_{abc}$ and S_{abc} is not a subset of S_{ab}, we replace S_{ab} and S_{abc} by $S_{ab\cup abc}$, i.e., S_{abc}. These comparisons proceed to the following termsets analogously, until we compare S_{abcd} with S_{ac}. Since $lS_{abcd} \neq lS_{ac}$, we add S_{ac} to $\mathcal{C}$. This process continues until there are no termsets in the set of frequent

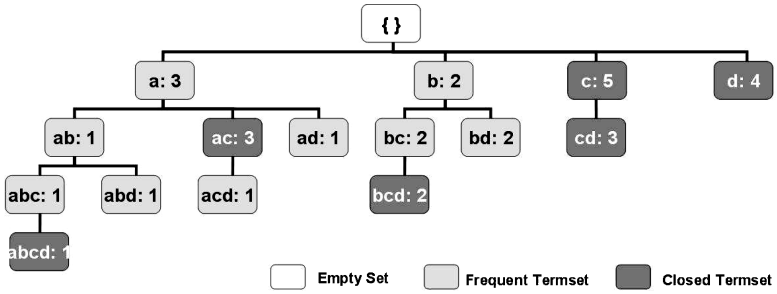


Fig. 2. Frequent and closed termsets for the sample document collection of Example 3.1 for all valid minimal frequency values.

termsets to be evaluated. Figure 2 shows the lattice of the frequent termsets with the closed ones highlighted.

4. SET-BASED VECTOR MODEL

To use termsets for ranking purposes, we propose a variant of the classic vector space model. This new IR model is referred to as *set-based vector model*, or simply set-based model. In this section, we discuss its fundamental features and its ranking algorithm.

4.1 Documents and Queries Representations

A document d_j and a user query q are represented as vectors of weighted termsets as follows:

$$\vec{d}_j = (w_{S_1,j}, w_{S_2,j}, \ldots, w_{S_{2^t},j})$$
$$\vec{q} = (w_{S_1,q}, w_{S_2,q}, \ldots, w_{S_{2^t},q})$$

where t corresponds to the number of distinct terms in the collection, $w_{S_i,j}$ is the weight of termset S_i in the document d_j, and $w_{S_i,q}$ is the weight of termset S_i in the query q.

One important simplification in our model is that the vector space is built just for the termsets generated from the query terms, i.e., documents and queries are represented by vectors in a 2^n-dimensional space, where n is the number of unique index terms in the user query. This is important because the number of termsets induced by the queries is usually small (see Section 9). Also, we can use proximity information among patterns of term co-occurrence to further reduce the number of termsets to be considered. Proximity information, as we shall see in Section 8, further provides significant gains in retrieval effectiveness.

4.2 Termset Weighting Schema

Good term-weighting schemes are usually based on three basic criteria. First, they take into account the number of times that an index term occurs in a document. Second, they emphasize term scarcity by reducing the weight of terms that appear in many documents. Third, they penalize long documents because these are naturally more likely to contain any given query term. This is usually

done with the introduction of a normalization factor to discount the contribution of long documents [Salton and Buckley 1988; Robertson et al. 1995].

Term weights can be calculated in many different ways [Salton and Yang 1973; Yu and Salton 1976; Salton and Buckley 1988]. To abide by the first two basic criteria above, the best-known term-weighting schemes use weights that are a function of (i) $tf_{i,j}$, the number of times that an index term i occurs in a document j, and (ii) df_i, the number of documents of the collection that contain an index term i. Since scarce terms are more selective, an inverse function of the document frequency df_i is used, the inverse document frequency idf_i. The resulting term-weighting strategy is called a $tf \times idf$ scheme.

In the set-based model, weights are associated with termsets (instead of terms). These weights are a function of the number of occurrences of the termset in a document and in the whole collection, analogously to $tf \times idf$ term weights. Formally, the weight of a termset S_i in a document d_j is defined as:

$$w_{S_i,j} = f(Sf_{i,j}) \times idS_i = (1 + \log Sf_{i,j}) \times \log\left(1 + \frac{N}{dS_i}\right), \tag{1}$$

where N is the number of documents in the collection, $Sf_{i,j}$ is the number of occurrences of the termset S_i in the document d_j and idS_i is the inverse frequency of occurrence of the termset S_i in the collection, scaled down by a log function. The factor $Sf_{i,j}$ subsumes $tf_{i,j}$ in the sense that it counts not only single terms but also co-occurring term subsets. The component idS_i subsumes the idf_i factor.

Example 4.1. Consider the dataset and the query of Example 3.1. Let us focus on the termset S_{ac} and its inverted list $lS_{ac} = \{d_1, d_3, d_5\}$. The number of occurrences of S_{ac} in documents d_1, d_3 and d_5 is 2, 3 and 1, respectively. The weights for the termset S_{ac} and its associated documents are computed as follows:

$$w_{S_{ac},d_1} = (1 + \log 2) \times \log\left(1 + \frac{6}{3}\right) = 1.860112$$

$$w_{S_{ac},d_3} = (1 + \log 3) \times \log\left(1 + \frac{6}{3}\right) = 2.305561$$

$$w_{S_{ac},d_5} = (1 + \log 1) \times \log\left(1 + \frac{6}{3}\right) = 1.098612.$$

Thus, the termset S_{ac} is more important for characterizing the document d_3.

4.3 Ranking Computation

In the set-based model, we compute the similarity between a document and the user query as the normalized scalar product between the document vector $\vec{d}_j$, $1 \leq j \leq N$, and the query vector $\vec{q}$, as follows:

$$sim(q, d_j) = \frac{\vec{d}_j \bullet \vec{q}}{|\vec{d}_j| \times |\vec{q}|} = \frac{\sum_{S_i \in S_q} w_{S_i,j} \times w_{S_i,q}}{|\vec{d}_j| \times |\vec{q}|}, \tag{2}$$

where $w_{S_i,j}$ is the weight associated with the termset S_i in the document d_j, $w_{S_i,q}$ is the weight associated with the termset S_i in the query q, and S_q is the set of all termsets generated from the query terms. That is, our ranking computation is restricted to the termsets generated by the query.

The norm of d_j, represented as $|\vec{d}_j|$, is hard to compute because of the large number of termsets generated by a document. To speed up computation, we consider only the 1-termsets in the document, i.e., we use only single terms. Thus, our normalization procedure does not take into account term co-occurrences. Despite that, it addresses the third ranking criterion in Section 4.2 because it accomplishes the effect of penalizing large documents, the major objective of ranking normalization. We validate this 1-termset normalization procedure through experimentation (see Section 7.3).

Example 4.2. In the dataset of the Example 3.1 consider the closed termsets S_{ac} and S_c and the document d_1. The weights for the termsets S_{ac} and S_c with respect to the document d_1, obtained using the Eq. (1), have values of 1.860112 and 1.334974, respectively. Let us use as normalization factor the norm of the document d_1, which is equal to 2.466123. Then, the similarity between the document d_1 and the query $q = \{a, b, c, d\}$, computed according to the set-based model, is given by:

$$sim(\{a, b, c, d\}, d_1) \approx \frac{w_{S_{ac},d_1} + w_{S_c,d_1}}{|d_1|} = \frac{1.860112 + 1.334974}{2.466123} = 1.295590,$$

where we did not take into account $|q|$ because it is a common factor to all documents.

To compute the ranking with regard to a user query q, we use the algorithm of Figure 3. First, we initialize the data structures (line 4) used for computing partial similarities between each termset S_i and a document d_j. For each query term, we retrieve its inverted list and determine the frequent termsets of size 1, applying the minimal frequency threshold mf (lines 5 to 10). The next step is the enumeration of all termsets based on the 1-termsets, filtered by the minimal frequency and proximity thresholds (line 11). After enumerating all termsets, we compute the partial similarity of each termset S_i with regard to the document d_j (lines 12 to 17). Following, we normalize the document similarities A by dividing each document similarity A_j by the norm of the document d_j (line 18). The final step is to select the k largest similarities and return the corresponding documents (line 19).

4.4 Indexing Data Structures and Algorithm

The index structure used by the set-based model corresponds to the compressed inverted files [Witten et al. 1999]. A general inverted file index for a text collection consists of two main components: (i) a set of inverted file entries, one entry per index term, each entry composed of the identifiers of the documents containing the corresponding index term, an intra-document frequency, and, optionally, a list of ordinal positions at which the term occurs in the document; and (ii) a data structure for identifying the location of the inverted file entry for each term, composed of a vocabulary of query terms and of an index mapping

SBM $(\mathbf{q}, \mathbf{mf}, \mathbf{mp}, \mathbf{k})$
$\mathbf{q}$ *: a set of query terms*
$\mathbf{mf}$ *: minimum frequency threshold*
$\mathbf{mp}$ *: minimum proximity threshold*
$\mathbf{k}$ *: number of documents to be returned*
1. *Let* $\mathbf{A}$ *be a set of accumulators*
2. *Let* $\mathbf{C_q}$ *be a set of 1-termsets*
3. *Let* $\mathbf{S_q}$ *be a set of termsets*
4. $A = \emptyset, \quad C_q = \emptyset, \quad S_q = \emptyset$
5. *for each query term* $t \in q$ *do begin*
6. *if* $df_t \geq mf$ *then begin*
7. *Obtain the 1-termset* S_t *from term* t
8. $C_q = C_q \ \cup \ \{S_t\}$
9. *end*
10. *end*
11. $S_q = Termsets\_Gen\,(C_q, mf, mp)$
12. *for each termset* $S_i \in S_q$ *do begin*
13. *for each* $[d_j, Sf_{i,j}]$ *in* lS_i *do begin*
14. *if* $A_j \notin A$ *then* $A = A \cup \{A_j\}$
15. $A_j = A_j + w_{S_i,j} \times w_{S_i,q}$, *from Eq. (1).*
16. *end*
17. *end*
18. *for each accumulator* $A_j \in A$ *do* $A_j = A_j \div |\vec{d_j}|$
19. *determine the* k *largest* $A_j \in A$ *and return the corresponding documents*
20. *end*

Fig. 3. The set-based model ranking algorithm.

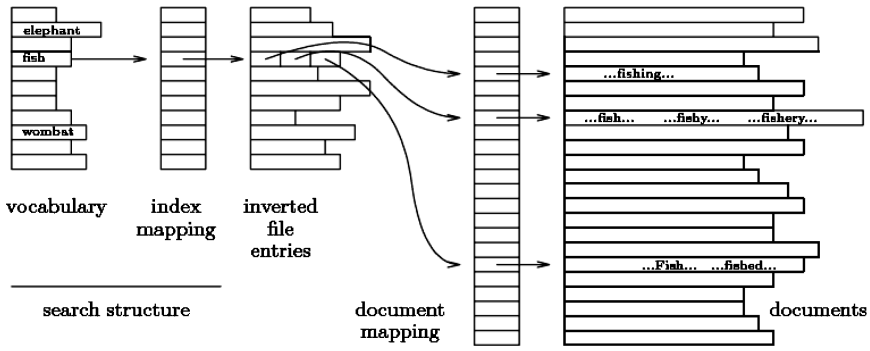


Fig. 4. The inverted file index structure.

(that maps ordinal term numbers onto disk addresses in the inverted index). This arrangement is illustrated in Figure 4. The index mapping can either be stored on disk as a separate file or can be held in memory with the vocabulary. We assume that inverted file entries store ordinal document numbers rather than addresses. Thus, to map the resulting document identifiers to disk addresses there must also be a document mapping.

We store the contents of each inverted file entry contiguously, in contrast to other schemes in which entries are often stored as linked lists with each node randomly placed on disk. Queries are processed according to the ranking algorithm presented, discussed in the previous section, which uses the vocabulary

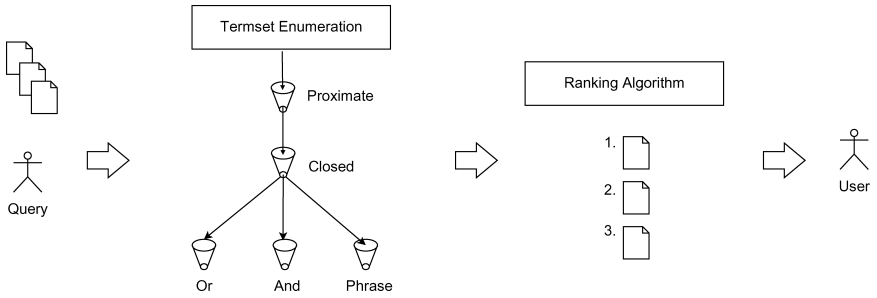


Fig. 5. The set-based model work-flow.

and index mapping to find the location of the inverted file entry for each query term.

We use a sort-based compressed multi-way merge algorithm, extensively discussed in Witten et al. [1999], to build the index structure for the 1-termsets. The inverted lists of high-order termsets, that is, termsets with more than one term, are build during query processing and do not have to be stored on disk.

5. QUERY PROCESSING

In Boolean retrieval systems [Buell 1981; Paice 1984], the terms in the user query are connected by the Boolean operators AND, OR and NOT. Boolean connectives are useful for specialized users who, knowing the document collection well, can use them to provide a more selective structure to their queries.

Figure 5 shows the set-based model work-flow. The first step consists in the specification of a user query. Next, the set-based model enumerates all closed termsets according to the query type (disjunctive, conjunctive and phrase queries) and the frequency and proximity thresholds. As we shall see in detail in the following sections, the evaluation of the enumerated closed termsets is quite different depending on the query type being considered. Finally, the documents are ranked according to their similarities to the enumerated termsets. The ranked documents are returned to the user.

5.1 Disjunctive Queries

One of the main advantages of the vector space model is its partial matching strategy, which allows the retrieval of documents that approximate the query conditions. This strategy corresponds, conceptually, to the processing of disjunctive queries.

Given a user query, the minimal frequency, and the proximity thresholds, the enumeration algorithm determines all closed termsets. Since the closed termsets represent all query-related patterns of term co-occurrence, partial matching between the query and the documents is allowed.

Example 5.1. Consider our sample collection of Example 3.1 and the user query $q = \{a, b, c, d\}$. Assume that the minimal frequency and minimal proximity threshold values are set to 1 and 10, respectively. Then, the termsets

enumeration algorithm finds six closed termsets associated with q, S_c, S_d, S_{ac}, S_{cd}, S_{bcd}, and S_{abcd} (see Figure 2), all of which occur in our sample collection.

5.2 Conjunctive Queries

Different search engines and portals might have different default semantics for handling a multiword query. Despite that, all major search engines assume conjunctive queries as a default querying strategy. That is, all query words must appear in a document that is included in the ranking.

The main modification of the set-based model for the processing of conjunctive queries is related to the termset enumeration algorithm. Since all query terms must occur in a document retrieved, we check if the document includes a closed termset that contains all query terms. If so, just the inverted list of these closed termset is evaluated by our ranking algorithm. Another important constraint is related to the minimal frequency threshold. We set this threshold to 1 because all documents containing all the query terms must be returned.

Example 5.2. We use the same dataset of Example 3.1, where $q = \{a, b, c, d\}$ and C is the collection of documents. We first check if the set of closed termsets contains a termset that has all query terms. In this simple example, this is the termset S_{abcd}. Its inverted list is then evaluated using our ranking algorithm. As a result, the document d_5 is returned.

5.3 Phrase Queries

A fraction of the queries in the Web include phrases, that is, a sequence of terms enclosed in quotation marks, which means that the phrase must appear in the documents retrieved. A standard way to evaluate phrase queries is to use an extended index that includes information on the positions at which a term occurs in a document. Given information on the positions of the terms, we can determine which documents contain a phrase declared in a query.

The set-based model can be easily adapted to handle phrase queries. To achieve this, we enumerate the set of closed termsets using the same restrictions applied for conjunctive queries. If there is a closed termset containing all query terms, we just need to verify if the query terms are adjacent. This is done by checking whether the ordinal word positions in the index are adjacent. The minimal proximity threshold is set to 1 to select only the adjacent termsets.

Example 5.3. Consider again the dataset of Example 3.1, where $q = \{$“a b c d”$\}$. The closed termset S_{abcd} matches the requeriments for phrase query processing. Thus, just its inverted list is evaluated by the ranking algorithm.

6. EXPERIMENTAL SETUP

In this section, we describe our experimental environment. We present the metrics and the reference collections.

6.1 Metrics

In this evaluation, we use two reference collections. Each of them is composed by a set of documents, a set of example queries, and a set of relevant responses

Table III. Characteristics of the Two Reference Collections TREC-8 and WBR99

Characteristics	Collection	
	TREC-8	WBR99
Number of Documents	528,155	5,939,061
Number of Distinct Terms	737,833	2,669,965
Number of Available Topics	450	100,000
Number of Topics Used	50 (401–450)	50
Avg. Terms per Topic (OR)	10.80	1.94
Avg. Terms per Topic (AND and PHRASE)	4.38	1.94
Avg. Relevants per Topic	94.56	35.40
Size (GB)	2	16

for each example query. We quantify the retrieval effectiveness of the various approaches through standard measures of average recall and precision. Computational efficiency is evaluated through query response times.

We have employed four aggregate metrics for measuring retrieval effectiveness: (i) standard 11-point average precision figures, (ii) average precision over the retrieved documents, (iii) average precision at 10, that is, average precision computed over the first ten documents retrieved, and (iv) document level averages, which corresponds to the precision at nine cutoff values.

Measuring differences in precision and recall between retrieval systems is only indicative of the relative performance. It is also necessary to establish whether the difference is statistically significant. Per-query recall-precision figures can be used in conjunction with statistical significance tests to establish the likelihood that a difference is significant. We use the "Wilcoxon's rank test", which has been shown by Zobel [1998] (and others) to be suitable for this task. In our comparisons, a 95% level of confidence is used to find whether the results are statistically significant.

6.2 The Reference Collections

In our evaluation, we use two reference collections WBR99 and TREC-8 [Voorhees and Harman 1999]. Table III presents the main features of these collections.

6.2.1 *TREC-8.* The TREC collections have been growing steadily over the years. At TREC-8 [Voorhees and Harman 1999], which is used in our experiments, the collection size is roughly 2 gigabytes (disks 4 and 5, excluding the Congressional Record subcollection). The documents in the TREC-8 collection come from the following sources: The Financial Times (1991–1994), Federal Register (1994), Foreign Broadcast Information Service, and LA Times.

Each TREC collection includes a set of example *information requests* that can be used for testing a new ranking algorithm. Each information request is a description of an information need in natural language. The TREC-8 has a total of 450 such requests, usually referred to as topics. Our experiments are performed with the 50 topics numbered 401–450. All queries were generated automatically in the following way: the disjunctive queries were generated using the title, description and narrative of each topic and the conjunctive

and phrase queries were generated using just the title and description of the topics. Disjunctive query generation is different from conjunctive and phrase queries generation because the latter ones require that the terms represent valid relationships or phrases. The mean number of keywords per query is 10.80 for disjunctive query processing and 4.38 for conjunctive and phrase query processing.

In TREC-8, the set of relevant documents for each information request was obtained as follows. For each information request, a pool of candidate documents was created by taking the top k documents in the ranking generated by various retrieval systems participating in the TREC conference. The documents in the pool were then shown to human assessors who ultimately decided on the relevance of each document. The average number of relevant documents per topic is 94.56.

6.2.2 *WBR99.* The WBR99 reference collection is composed of a database of Web pages, a set of example Web queries, and a set of relevant documents associated with each example query. The database is composed of 5,939,061 pages of the Brazilian Web, under the domain ".br". The pages were automatically collected by the document crawler described in Silva et al. [1999].

A total of 50 example queries were selected from a log of 100,000 queries submitted to the *TodoBR* search engine.[1] The queries selected were the 50 most frequent ones, excluding queries related to sex. The mean number of keywords per query is 1.94 (disjunctive, conjunctive, and phrase queries). Of the 50 selected queries, 28 were quite general, like "tabs", "movies", or "mp3". Other 14 queries were more specific, but still on a general topic, like "transgenic food" or "electronic commerce". Finally, eight queries were quite specific, consisting mainly of music band names. Similarly to Hawking and Craswell [2001], for all queries, a description of what documents should be considered relevant was shown to the users. For instance, for the query "employment", users were instructed to consider relevant only sites dedicated to employment advertisements.

Although this is a very small percentage of the total number of queries submitted to the TodoBR search engine, and although the set of most frequent queries might change with time due to shifts in the interest of the users, the selected queries represent typical Web querying behavior in the sense that they are short, imprecise, and cover a wide variety of topics, as reported in Spink et al. [2002].

For each of our 50 example queries, we composed a query pool formed by the top 20 documents, retrieved by each of the eight ranking variants we considered, that is, disjunctive queries with the vector, the set-based, and the proximity set-based models; conjunctive queries with the vector, the set-based, and the proximity set-based models; phrase queries with the vector and the set-based models. Each query pool contained an average of 83.26 pages. All documents in each query pool were submitted to a manual evaluation by a group of 10 users, all of them familiar with Web searching and with a Computer Science background. Users were allowed to follow links and to evaluate the pages according

[1]http://www.todobr.com.br.

not only to their textual content, but also according to their linked pages and graphical content (flash, or dynamic HTML animations). The average number of relevant pages per query pool is 35.4. We adopted the same pooling method used for the Web-based collection of TREC [Hawking et al. 1998, 1999].

7. TUNING OF THE SET-BASED MODEL

In this section, we show experimental results for tuning the set-based model (SBM) and the proximity set-based model (PSBM), that is, the set-based model using proximity information. Our analysis verifies the impact of frequency and proximity thresholds in the retrieval effectiveness, and also presents a normalization evaluation using four well-known normalization techniques.

For each test collection evaluated, TREC-8 and WBR99, we built a training set composed of 15 queries, randomly selected from the 50 available sample queries. These training sets were used for tuning of the minimal frequency and of the minimal proximity thresholds. They were also used for selecting a normalization technique among the four we had available.

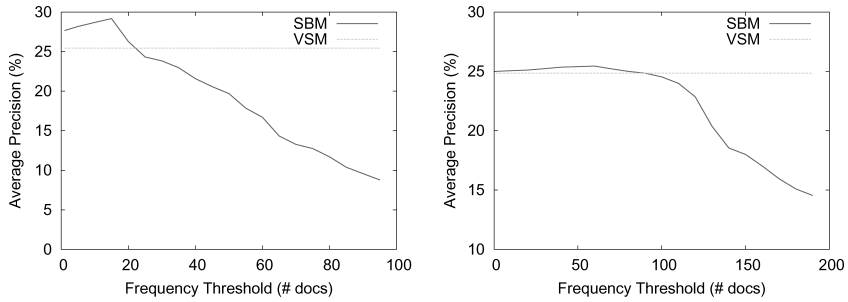
Our experiments were performed in a Linux-based PC with a AMD-athlon 2600 + 2.0 GHz processor and 512 MBytes of RAM. They used the other 35 queries in the set of 50 sample queries. The detailed results are reported in Section 8.

7.1 Minimal Frequency Evaluation

One of the key features of the set-based model is the possibility of controlling the minimal frequency threshold. By varying the minimal frequency, we can exploit a trade-off between precision and the number of termsets taken into consideration. The higher the minimum frequency, the smaller the number of termsets to consider, and faster the computation. However, the impact on average precision needs to be analyzed.

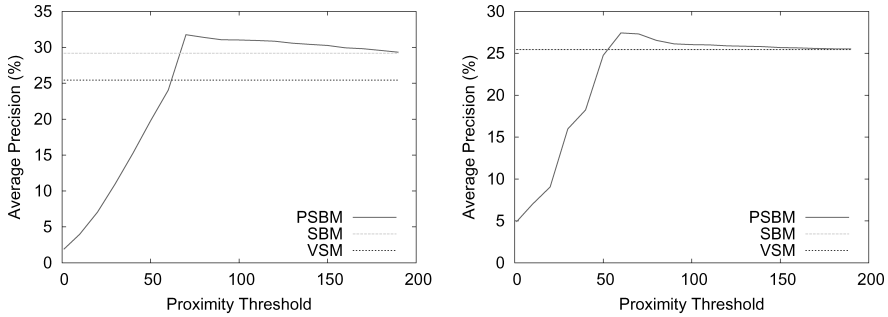
To illustrate, we performed a series of executions of the set-based model with the 15 disjunctive queries in our training set. For the TREC-8 test collection, we varied the minimal frequency threshold from 1 to 90 documents. For the WBR99 test collection, we varied it from 1 to 190 documents. The results are presented in Figure 6. We observe that the set-based model average precision is significantly affected by the minimal frequency threshold, and the behavior is similar for both collections. Initially, an increase in the minimal frequency results in better precision. Maximum precision is reached for a minimal frequency threshold equal to 15 documents for the TREC-8 and 60 for the WBR99. These are the values used in our experimentation in Section 8.

For larger threshold values, precision decreases as the threshold value increases. This behavior can be explained as follows: First, an increase in the minimal frequency causes irrelevant termsets to be discarded, resulting in better precision. When the minimal frequency becomes larger than its best value, relevant termsets start to be discarded, leading to a reduction in overall precision.



(a) TREC-8 results. (b) WBR99 results.

Fig. 6. Impact on average precision of varying the minimal frequency threshold for the set-based model (SBM) and the standard vector space model (VSM), in the TREC-8 and the WBR99 test collections. A training set of 15 queries was used with each collection.



(a) TREC-8 results. (b) WBR99 results.

Fig. 7. Impact on average precision of varying the minimal proximity threshold for the proximity set-based model (PSBM), for the set-based model (SBM) and the standard vector space model (VSM), in the TREC-8 and the WBR99 test collections. A training set of 15 queries was used with each collection.

7.2 Minimal Proximity Evaluation

We also verified how variations in the minimal proximity threshold affect average precision. We performed a series of executions of the proximity set-based model with the 15 disjunctive queries in our training set in which the minimal proximity threshold was varied from 1 to 190 for both test collections. The results are illustrated in Figure 7. We observe that the proximity set-based model is significantly affected by the minimal proximity threshold, and the behavior is quite similar for both collections. Initially, an increase in the minimal proximity results in better precision, with maximum precision being achieved for minimal proximity values of 70 and 60 words for the TREC-8 and WBR99 collections, respectively. These are the values used in our experimentation in Section 8.

When we increase the minimal proximity beyond those values, we observe that the precision decreases until it reaches almost the mean average precision obtained by the set-based model. This behavior can be explained as follows: First, an increase in the minimal proximity implies an increase in the number of termsets evaluated, which yields better precision. When the minimal proximity

increases beyond the best found values, the number of termsets representing weak correlations (i.e., correlations among terms that are separated far apart from each other) increases, leading to a reduction in average precision figures.

7.3 Normalization Evaluation

Long and verbose documents include repeated references to a same term. As a result, term frequencies tend to be higher for these documents. This increases the likelihood that a long document will be retrieved by a user query. To avoid this undesirable effect, document length normalization is used. It provides a way of penalizing long documents. Various normalization techniques have been used in information retrieval systems, especially with the standard vector space model.

For efficiency reasons, the normalization in the set-based model is based only on 1-termsets. That is, the first order termsets, the most influential, are considered for normalization. This is important because computing the norm of a document using closed termsets might be prohibitively costly.

We discuss experimental results of applying four popular term-based normalization techniques to the set-based model. These normalization techniques are as follows.

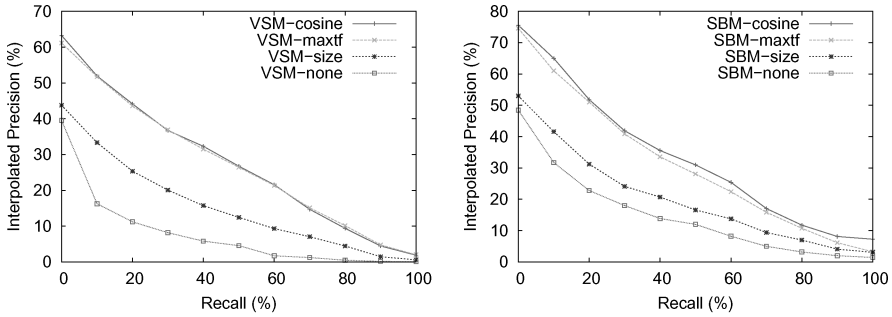
Technique 7.1. Cosine normalization is the most commonly used normalization technique in the vector space model. In our case, every termset weight in a document is divided by the Euclidean weighted norm of the document vector. The cosine normalization factor for a document j is computed as $\sqrt{w_{S_1,j}^2 + w_{S_2,j}^2 + \cdots + w_{S_t,j}^2}$, where $w_{S_i,j}$ is the weight of 1-termset S_i in document d_j. Termsets absent from a document are considered to have zero weight.

Technique 7.2. Another popular normalization technique is to first normalize individual *tf* weights by the maximum *tf*. The SMART retrieval system [Salton and Buckley 1988] augmented *tf* factor, given by $0.5 + 0.5 \times \frac{tf}{max_{tf}}$, and the *tf* weights used in the INQUERY system [Turtle and Croft 1990], given by $0.4 + 0.6 \times \frac{tf}{max_{tf}}$, are examples of such normalization. Using normalized *tf* coefficients, we apply the cosine normalization factor given by $\sqrt{w_{S_1,j}^2 + w_{S_2,j}^2 + \cdots + w_{S_t,j}^2}$.

Technique 7.3. More recently, a length normalization scheme has been used in the Okapi system [Robertson et al. 1995], which is based on the byte size of the documents. This scheme does not introduce mutual dependences among the term weights in a document.

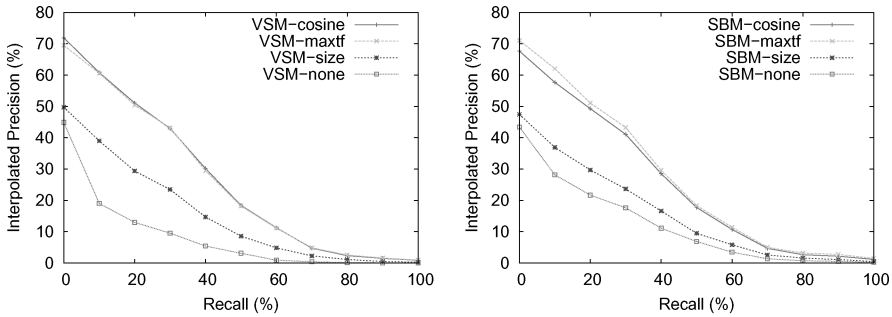
Technique 7.4. We also evaluated the retrieval effectiveness of the vector space and set-based models, when no normalization scheme was used.

The experimental results for the vector space model and the set-based model models using Cosine, Maxtf, Size, and None normalization techniques for processing the 15 queries in our training set are depicted in Figures 8 and 9. We



(a) Vector space model (VSM) results. (b) Set-based model (SBM) results.

Fig. 8. Normalization recall-precision curves for the TREC-8 collection using a training set of 15 queries.



(a) Vector space model (VSM) results. (b) Set-based model (SBM) results.

Fig. 9. Normalization recall-precision curves for the WBR99 collection using a training set of 15 queries.

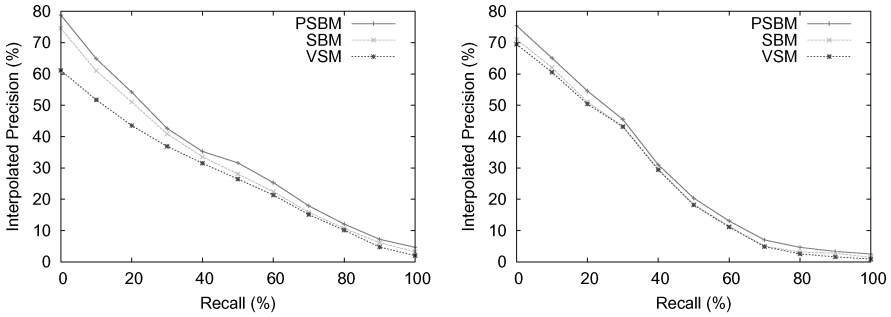
observe that the effects of the four techniques are analogous in all cases considered. Our results indicate that Maxtf is the method of choice. Thus, we adopt the Maxtf normalization technique in all our experiments.

8. RETRIEVAL PERFORMANCE

In this section, we show the retrieval performance for the set-based model and the proximity set-based model. Our analysis is based on a comparison to the standard vector space model using the 35 queries in our test set.

8.1 Disjunctive Queries

Figure 10 shows the 11-point average precision figures for the set-based model, the proximity set-based model, and the vector space model algorithms. We observe that the set-based model and the proximity set-based model yield better precision than the vector space model, regardless of the collection and of the recall level. The proximity set-based model ranking yields the largest improvements, which suggests that proximity information can be of value. Further, we observe that the improvements are larger for the TREC-8 collection, because its larger queries allow computing a more representative set of closed termsets.



(a) TREC-8 results. (b) WBR99 results.

Fig. 10. Precision-recall curves for the vector space model (VSM), the set-based model (SBM), and the proximity set-based model (PSBM) when disjunctive queries are used, with the TREC-8 and the WBR99 test collections, using the test set of 35 sample queries.

Table IV. TREC-8 Document Level Average Figures for the Vector Space Model (VSM), the Set-Based Model (SBM), and the Proximity Set-Based Model (PSBM) with Disjunctive Queries

TREC-8—Disjunctive Queries					
	Precision (%)			Gain (%)	
Level	VSM	SBM	PSBM	SBM	PSBM
At 5 docs	46.50	61.45	63.87	32.15	37.35
At 10 docs	43.75	56.10	59.19	28.22	35.29
At 15 docs	40.23	50.74	52.72	26.12	31.04
At 20 docs	38.45	47.33	49.82	23.09	29.57
At 30 docs	35.61	42.41	44.72	19.09	25.58
At 100 docs	24.41	26.45	29.17	08.35	19.50
At 200 docs	17.91	18.82	20.99	05.08	17.19
At 500 docs	10.25	10.26	11.61	00.09	13.26
At 1000 docs	05.19	05.29	05.76	01.92	10.98
Average	25.44	29.17	31.76	14.66	24.84

Detailed average precision figures are presented in Tables IV and V. Let us examine first the results for the TREC-8 collection. At the top 10 documents, the proximity set-based model provided a very significant gain in average precision of 35.29%, relative to the vector space model. This result clearly shows that termset information can be used to considerably improve retrieval effectiveness. The set-based model, which disposes with information on proximity, also provides a significant gain of 28.22%.

Let us examine now the results for the WBR99 collection. At the top 10 ranked documents, the gains in average precision were of 2.76% for the set-based model and of 10.66% for the proximity set-based model. That is, with very short queries termset information is not very useful because the number of termsets in the user query is too small. Even though, if proximity information is factored in, consistent gains in average precision are observed.

Detailed average precision difference comparison is presented in Table VI. This table summarizes two distinct results. The first result is the count of queries where there is a difference between two retrieval techniques. This is

Table V. WBR99 Document Level Average Figures for the Vector Space Model (VSM), the Set-Based Model (SBM), and the Proximity Set-Based Model (PSBM) with Disjunctive Queries

WBR99—Disjunctive Queries					
	Precision (%)			Gain (%)	
Level	VSM	SBM	PSBM	SBM	PSBM
At 5 docs	47.99	50.03	54.00	04.25	12.52
At 10 docs	45.21	46.46	50.03	02.76	10.66
At 15 docs	41.15	41.55	45.47	00.97	10.49
At 20 docs	39.12	39.47	42.52	00.89	08.69
At 30 docs	35.99	36.14	38.01	00.41	05.61
At 100 docs	21.79	22.01	23.10	00.55	06.01
At 200 docs	12.58	12.92	13.22	01.91	05.08
At 500 docs	08.55	08.79	09.02	01.64	05.49
At 1000 docs	03.16	03.25	03.35	01.27	06.01
Average	24.85	25.44	37.43	02.37	10.38

Table VI. Comparison of Average Precision of the Vector Space Model (VSM), the Set-Based Model (SBM), and the Proximity Set-Based Model (PSBM) with Disjunctive Queries

Disjunctive Queries			
	Statistical Significance		
Collection	SBM/VSM	PSBM/VSM	PSBM/SBM
TREC-8	58/18	79/11	64/20
WBR99	40/15	61/10	55/26

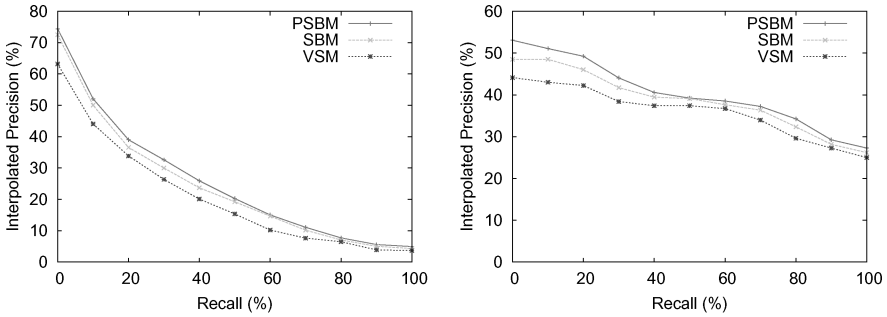
Each entry has two numbers X and Y (that is, X/Y). X is the percentage of queries where a technique A is better that a technique B. Y is the percentage of queries where a technique A is worse than a technique B. The numbers in bold represent the significant results using the "Wilcoxon's signed rank test" with a 95% confidence level.

expressed in terms of the proportion of queries that differ. The second result is the test for the statistical significance; significant differences are shown in bold.

Retrieval based on the set-based and the proximity set-based models was found to be significantly better than the vector space model for the TREC-8 test collection. For the WBR99 test collection, the distinction between set-based model and the standard vector space model is not clear. There is a significant difference between the proximity set-based model and the other evaluated models. However, no significant differences was found when the set-based model is directly compared with the vector space model. That is, with very short queries termset information is not very useful because the number of termsets in the user query is too small, and the majority of the enumerated termsets corresponds to 1-termsets.

8.2 Conjunctive Queries

We now discuss the precision-recall results when conjunctive query processing is considered. The minimal frequency threshold is set to 1. The minimal proximity threshold values are set according to the tuning discussed in Section 7.2.



(a) TREC-8 results. (b) WBR99 results.

Fig. 11. Precision-recall curves for the vector space model (VSM), the set-based model (SBM), and the proximity set-based model (PSBM) when conjunctive queries are used, with the TREC-8 and the WBR99 test collections, using the test set of 35 sample queries.

Table VII. TREC-8 Document Level Average Figures for the Vector Space Model (VSM), the Set-Based Model (SBM), and the Proximity Set-Based Model (PSBM) with Conjunctive Queries

TREC-8—Conjunctive Queries					
	Precision (%)			Gain (%)	
Level	VSM	SBM	PSBM	SBM	PSBM
At 5 docs	46.71	60.45	62.47	29.42	33.74
At 10 docs	43.80	55.90	57.29	27.63	30.80
At 15 docs	39.98	50.13	51.92	25.39	29.86
At 20 docs	35.02	44.53	45.79	27.16	30.75
At 30 docs	33.23	41.12	43.16	23.74	29.88
At 100 docs	22.12	25.95	28.02	17.31	26.67
At 200 docs	13.45	15.95	16.99	18.59	26.32
At 500 docs	7.78	9.07	9.81	16.58	26.09
At 1000 docs	3.56	4.09	4.40	14.89	25.28
Average	19.96	23.23	25.94	16.38	29.96

As we can see in Figure 11, the set-based model and the proximity set-based model yield better precision than the vector space model, regardless of the collection and of the recall level. As in the case of disjunctive queries, the proximity set-based model yields the highest results. Also, as before, the improvements are larger in the TREC-8 collection because its larger queries allow computing a more representative set of closed termsets.

Accounting for correlations never degrades the quality of the response sets. In fact, as presented in Tables VII and VIII, the set-based model yields improvements in average precision of 16.38% and 7.29% for the TREC-8 and the WBR99 collections, respectively. For the proximity set-based model the improvements are of 29.96% and 11.04% for the TREC-8 and the WBR99 collections, respectively.

At the top 10 ranked documents, the gains in precision are higher. For instance, the gains in average precision were 7.79% (WBR99) and 27.63% (TREC-8) for the set-based model, and 11.89% (WBR99) and 30.80% (TREC-8) for the proximity set-based model.

Table VIII. WBR99 Document Level Average Figures for the Vector Space Model (VSM), the Set-Based Model (SBM), and the Proximity Set-Based Model (PSBM) with Conjunctive Queries

WBR99—Conjunctive Queries					
	Precision (%)			Gain (%)	
Level	VSM	SBM	PSBM	SBM	PSBM
At 5 docs	46.71	50.45	52.25	08.01	11.86
At 10 docs	43.91	47.33	49.13	07.79	11.89
At 15 docs	35.01	37.34	38.61	06.66	10.28
At 20 docs	31.53	33.61	34.99	06.60	10.97
At 30 docs	29.64	31.44	32.37	06.07	09.21
At 100 docs	28.76	30.01	30.85	04.35	07.27
At 200 docs	27.89	29.05	29.39	04.16	05.38
At 500 docs	22.61	23.22	23.99	02.70	06.10
At 1000 docs	21.99	22.62	22.97	02.86	04.46
Average	33.60	36.05	37.31	7.29	11.04

Table IX. Comparison of Average Precision of the Vector Space Model (VSM), the Set-Based Model (SBM), and the Proximity Set-Based Model (PSBM) with Conjunctive Queries

Conjunctive Queries			
	Statistical Significance		
Collection	SBM/VSM	PSBM/VSM	PSBM/SBM
TREC-8	**57/20**	80/12	65/24
WBR99	53/13	62/10	57/28

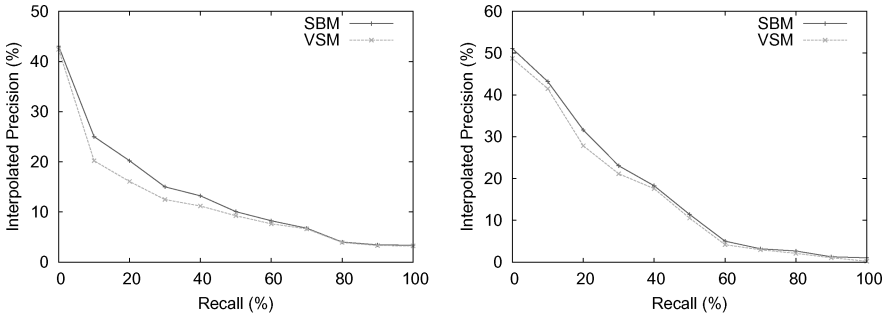
Each entry has two numbers X and Y (that is, X/Y). X is the percentage of queries where a technique A is better that a technique B. Y is the percentage of queries where a technique A is worse than a technique B. The numbers in bold represent the significant results using the "Wilcoxon's signed rank test" with a 95% confidence level.

Table IX shows the statistical significance tests for the evaluated models. The proximity set-based model was found to be significantly better than the vector space model and the set-based model for both test collections (TREC-8 and WBR99). The results found for the set-based model was significantly better than the standard vector space model only for the TREC-8 test collection. As mentioned before, the very short queries found in the WBR99 collection directly affects the results for the set-based model.

8.3 Phrase Queries

We now discuss our results when phrase query processing is used. In this case, the proximity set-based model corresponds to the set-based model, since the minimal proximity threshold must be used. The results were obtained by setting the minimal frequency and the minimal proximity thresholds to 1.

As we can see in Figure 12, the set-based model yields better precision than the vector space model, regardless of the collection and of the recall level. Tables X and XI detail the numeric figures. We observe that the set-based model yields improvements in average precision of 17.51% and 8.93% for the TREC-8 and the WBR99 collections, respectively. At the top 10 ranked documents,



(a) TREC-8 results. (b) WBR99 results.

Fig. 12. Precision-recall curves for the vector space model (VSM) and the set-based model (SBM) when phrase queries are used, with the TREC-8 and the WBR99 test collections, using the test set of 35 sample queries

Table X. Document Level Average Figures for the Vector Space Model (VSM) and the Set-Based Model (SBM) Relative to the TREC-8 Test Collection, when Phrase Queries Are Used

TREC-8—Phrase Queries			
	Precision (%)		
Level	VSM	SBM	Gain (%)
At 5 docs	32.33	38.76	19.89
At 10 docs	26.12	31.05	18.87
At 15 docs	20.34	24.01	18.04
At 20 docs	15.25	17.55	15.08
At 30 docs	12.29	13.89	13.02
At 100 docs	9.02	9.72	07.76
At 200 docs	6.46	6.93	07.28
At 500 docs	2.89	3.04	05.19
At 1000 docs	1.17	1.21	03.42
Average	11.59	13.62	17.51

the gains in precision are higher. For instance, for the top 10 documents, the gains in average precision were 9.92% (WBR99) and 18.87% (TREC-8) for the set-based model.

Table XII shows the statistical significance tests for the evaluated models. The set-based model was found to be significantly better than the vector space model and the set-based model for both test collections (TREC-8 and WBR99).

The set-based model corresponds to the first information retrieval model that exploits term correlations and term proximity effectively, providing gains in average precision, while keeping computational costs low, as we now discuss.

9. COMPUTATIONAL EFFICIENCY

In this section, we compare our models to the standard vector space model, when query response times are considered. This is important because one major limitation of existing models that account for term correlations is their computational cost. Several of these models cannot be applied to large or even

Table XI. Document Level Average Figures for the Vector Space Model (VSM) and the Set-Based Model (SBM) Relative to the WBR99 Test Collection, when Phrase Queries Are Used

WBR99—Phrase Queries			
	Precision (%)		
Level	VSM	SBM	Gain (%)
At 5 docs	45.88	50.66	10.42
At 10 docs	39.13	43.01	09.92
At 15 docs	28.85	31.58	09.46
At 20 docs	17.32	18.81	08.60
At 30 docs	15.69	16.93	07.90
At 100 docs	13.83	14.80	07.01
At 200 docs	8.49	9.04	06.48
At 500 docs	4.63	4.92	06.26
At 1000 docs	1.99	2.11	06.03
Average	15.11	16.46	08.93

Table XII. Comparison of Average Precision of the Vector Space Model (VSM) and the Set-Based Model (SBM) with Phrase Queries

Phrase Queries	
	Statistical Significance
Collection	SBM/VSM
TREC-8	60/28
WBR99	55/15

Each entry has two numbers X and Y (that is, X/Y). X is the percentage of queries where a technique A is better that a technique B. Y is the percentage of queries where a technique A is worse than a technique B. The numbers in bold represent the significant results using the "Wilcoxon's signed rank test" with a 95% confidence level.

mid-size collections, since their costs increase exponentially with the vocabulary size.

9.1 Disjunctive Queries

We determined the average number of closed termsets and the average inverted list sizes for the set-based model and the proximity set-based model for disjunctive query processing. The results, presented in Table XIII, show that the average case scenario is much better than the worst case one. For the TREC-8 collection, the average number of closed termsets per query is 581.89 for the set-based model, and 432.55 for the proximity set-based model. Notice that the number of closed termsets is smaller than the worst case $2^{\lceil q \rceil} = 2048$, where $q = 10.80$.

Table XIV displays the response time for disjunctive query processing. We also calculated the increase in response time for the set-based and the proximity

Table XIII. Average Number of Closed Termsets and Inverted List Sizes for the Vector Space Model (VSM), the Set-Based Model (SBM), and the Proximity Set-Based Model (PSBM)

	# Closed Termsets		Avg. Inverted List Size		
Collection	SBM	PSBM	VSM	SBM	PSBM
TREC-8	581.99	432.55	20,234	6,151	4,189
WBR99	5.31	4.89	304,101	90,639	66,023

Table XIV. Average Response Times and Response Time Increases for the Vector Space Model (VSM), the Set-Based Model (SBM), and the Proximity Set-Based Model (PSBM) for Disjunctive Query Processing

	Average Response Time (s)			Increase (%)	
Collection	VSM	SBM	PSBM	SBM	PSBM
TREC-8	0.2314	0.3174	0.9691	37.16	318.79
WBR99	0.1179	0.1406	0.6265	19.25	431.38

set-based models, when compared to the vector space model. We observe that the set-based model computes in times 19.25% and 37.16% larger than the vector space model for the WBR99 and the TREC-8 collections, respectively. These results confirm our complexity analysis, indicating that the set-based model is comparable to vector space model in terms of computational costs. The increases in processing time for the proximity set-based model are much larger, 318.79% for the TREC-8 and 431.38% for the WBR99.

We identify two main reasons for the relatively small increase in execution time for the set-based model. First, there is a small number of query related termsets in the reference collections. As a consequence, the inverted lists associated tend to be small (as presented in Table XIII) and are usually manipulated in main memory in our implementation. Second, we employ pruning techniques that discard irrelevant termsets early in the computation. The main reason for the increase in execution time for the proximity set-based model is the size of the positional index. For instance, while the TREC-8 inverted list file has approximately 300 megabytes, its positional inverted list file has approximately 800 megabytes.

We also evaluated the average increase in the response time of the set-based model as a function of the number of terms in the query. Figure 13 summarizes the results of our experiments using all the 100,000 queries of the WBR99 collection. The execution time of the set-based model is directly affected by the number of terms in the query for a given threshold. Increases in the number of terms results in increases in the overall execution time. In this case, the execution time is dominated by operations over the inverted lists of the termsets obtained. Both the number of termsets and the size of their inverted lists increase with the query size.

9.2 Conjunctive Queries

Table XV displays the response time and the increase in response time for the set-based model and the proximity set-based model, when compared to vector space model for conjunctive query processing. All 100,000 queries submitted to

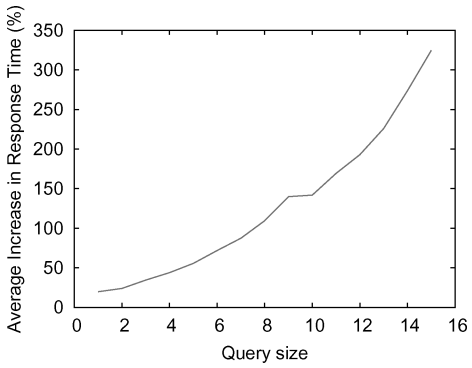


Fig. 13. Impact of query size on average response time in the WBR99 for the set-based model (SBM).

Table XV. Average Response Times and Response Time Increases for the Vector Space Model (VSM), the Set-Based Model (SBM), and the Proximity Set-Based Model (PSBM) for Conjunctive Query Processing

	Average Response Time (s)			Increase (%)	
Collection	VSM	SBM	PSBM	SBM	PSBM
TREC-8	0.0478	0.0637	0.1973	33.26	312.76
WBR99	0.0414	0.0501	0.1187	21.01	186.71

Table XVI. Average Response Times and Response Time Increases for the Vector Space Model (VSM) and the Set-Based Model (SBM) for Phrase Query Processing

	Avg. Response Time (s)		
Collection	VSM	SBM	Increase (%)
TREC-8	0.1073	0.1314	22.46
WBR99	0.1185	0.1399	18.05

the *TodoBR* search engine, excluding the unique term queries, were evaluated for the WBR99 collection. We observe that the set-based model computes in times 21.01% and 33.26% larger than the vector space model for the WBR99 and the TREC-8 collections, respectively. The increases in processing time for the proximity set-based model are much larger, 312.76% for TREC-8 and 186.71% for the WBR99.

9.3 Phrase Queries

Finally, the response time and the increase in response time for all models and collections considered for phrase query processing is presented in Table XVI. Only the phrase queries contained in the 100,000 queries submitted to the *TodoBR* search engine were evaluated for the WBR99 collection. We observe that the set-based model computes in times 18.05% and 22.46% larger than the vector space model for the WBR99 and the TREC-8 collections, respectively.

The execution time of both models, the set-based model and the vector space model, are dominated by operations over the positional inverted lists of the

query terms. The small increase in execution time for the set-based model corresponds to the closed termset enumeration phase.

10. CONCLUSIONS AND FUTURE WORK

We presented the set-based model, an information retrieval model that uses term correlations to compute term weights. Terms correlations are derived using association rules mining. We showed that our model allows significant improvement in retrieval effectiveness, as high as 30%, while keeping extra computational costs small, when compared to the standard vector space model.

We evaluated and validated our proposed information retrieval models using two test collections. Regarding retrieval effectiveness, we showed that our models are superior for all the collections and query types considered when compared to the standard vector space model. Further, the computational costs allow the use of our set-based model with general large collections.

For future work, we will use our association rules framework to structure queries as a set of smaller conjunctive subqueries. Information on the occurrences of the conjunctive subqueries in the document collection will be used to guide the query structuring process. We will also investigate how to apply the termsets framework to the probabilistic and statistical language models.

ACKNOWLEDGMENTS

We are most grateful to the anonymous referees for their very helpful comments.

REFERENCES

AGRAWAL, R., IMIELINSKI, T., AND SWAMI, A. 1993. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD International Conference Management of Data* (Washington, D.C.). P. Buneman and S. Jajodia, Eds. ACM, New York, 207–216.

AGRAWAL, R. AND SRIKANT, R. 1994. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Data Bases* (Santiago, Chile). J. B. Bocca, M. Jarke, and C. Zaniolo, Eds. Morgan-Kaufmann, San Matco, CA, 487–499.

ALSAFFAR, A. H., DEOGUN, J. S., RAGHAVAN, V. V., AND SEVER, H. 2000. Enhancing concept-based retrieval based on minimal term sets. *J. Intel. Inf. Syst. 14*, 2–3 (March–June), 155–173.

BAEZA-YATES, R. AND RIBEIRO-NETO, B. 1999. *Modern Information Retrieval*, 1st ed. Addison-Wesley-Longman, Wokingham, UK.

BERGER, A. AND LAFFERTY, J. 1999. Information retrieval as statistical translation. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Berkeley, CA). ACM, New York, 222–229.

BILLHARDT, H., BORRAJO, D., AND MAOJO, V. 2002. A context vector model for information retrieval. *J. Amer. Soc. Inf. Sci. Tech. 53*, 3, 236–249.

BOLLMANN-SDORRA, P., HAFEZ, A., AND RAGHAVAN, V. V. 2001. A theoretical framework for association mining based on the boolean retrieval model. In *Data Warehousing and Knowledge Discovery: Third International Conference* (Munich, Germany). Y. Kambayashi, W. Winiwarter, and M. Arikawa, Eds. Lecture Notes in Computer Science, vol. 2114. Springer-Verlag, New York, 21–30.

BOLLMANN-SDORRA, P. AND RAGHAVAN, V. V. 1998. On the necessity of term dependence in a query space for weighted retrieval. *J. Amer. Soc. Inf. Sci. 49*, 13 (Nov.), 1161–1168.

BUELL, D. 1981. A general model of query processing in information retrieval systems. *Inf. Proc. Manage. 17*, 249–262.

GAO, J., NIE, J., WU, G., AND CAO, G. 2004. Dependence language model for information retrieval. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and*

Development in Information Retrieval (Sheffield, South Yorkshire, UK). ACM, New York, 170–177.

HARPER, D. J. AND VAN RIJSBERGEN, C. J. 1978. An evaluation of feedback in document retrieval using co-occurrence data. *J. Document. 34*, 189–216.

HAWKING, D. AND CRASWELL, N. 2001. Overview of TREC-2001 web track. In *Proceedings of the Tenth Text REtrieval Conference (TREC-2001)*. E. M. Voorhees and D. K. Harman, Eds. Department of Commerce, National Institute of Standards and Technology, Gaithersburg, MD, 61–67.

HAWKING, D., CRASWELL, N., AND THISTLEWAITE, P. B. 1998. Overview of TREC-7 very large collection track. In *The Seventh Text REtrieval Conference (TREC-7)*. E. M. Voorhees and D. K. Harman, Eds. Department of Commerce, National Institute of Standards and Technology, Gaithersburg, MD, 91–104.

HAWKING, D., CRASWELL, N., THISTLEWAITE, P. B., AND HARMAN, D. 1999. Results and challenges in web search evaluation. *Comput. Netw. 31*, 11–16 (May), 1321–1330. Also in *Proceedings of the 8th International World Wide Web Conference*.

KASZKEIL, M. AND ZOBEL, J. 1997. Passage retrieval revisited. In *Proceedings of the 20th ACM SIGIR Conference on Research and Development in Information Retrieval* (Philadelphia, PA). ACM, New York, 178–185.

KASZKEIL, M., ZOBEL, J., AND SACKS-DAVIS, R. 1999. Efficient passage ranking for document databases. *ACM Trans. Inf. Syst. (TOIS) 17*, 4 (Oct.), 406–439.

KIM, M., ALSAFFAR, A. H., DEOGUN, J. S., AND RAGHAVAN, V. V. 2000. On modeling of concept based retrieval in generalized vector spaces. In *Proceedings of the International Symposium on Methods of Intelligent Systems* (Charlote, NC). Springer-Verlag, New York, 453–462.

LAFFERTY, J. AND ZHAI, C. 2001. Document language models, query models and risk minimization. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New Orleans, LA). ACM, New York, 111–119.

MARON, M. AND KUHNS, J. 1960. On relevance, probabilistic indexing and information retrieval. *J. ACM 7*, 216–244.

NALLAPATI, R. AND ALLAN, J. 2002. Capturing term dependencies using a language model based on sentence trees. In *Proceedings of the 11th International Conference on Information and Knowledge Management* (McLean, VA). ACM, New York, 383–390.

PAICE, C. D. 1984. Soft evaluation of boolean search queries in information retrieval systems. *Inf. Tech. 3*, 1, 33–41.

PONTE, J. M. AND CROFT, W. B. 1998. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Melbourne, Australia). ACM, New York, 275–281.

PÔSSAS, B., ZIVIANI, N., AND MEIRA, JR., W. 2002a. Enhancing the set-based model using proximity information. In *Proceedings of the 9th International Symposium on String Processing and Information Retrieval* (Lisbon, Portugal). Lecture Notes in Computer Science. Springer-Verlag, New York, 104–116.

PÔSSAS, B., ZIVIANI, N., MEIRA, JR., W., AND RIBEIRO-NETO, B. 2002b. Set-based model: A new approach for information retrieval. In *Proceedings of the 25th ACM-SIGIR Conference on Research and Development in Information Retrieval*. ACM Press, Tampere, Finland, 230–237.

PÔSSAS, B., ZIVIANI, N., RIBEIRO-NETO, B., AND MEIRA, JR., W. 2004. Processing conjunctive and phrase queries with the set-based model. In *Proceedings of the 11th International Symposium on String Processing and Information Retrieval* (Padova, Italy). Lecture Notes in Computer Science. Springer-Verlag, New York, 171–183.

RAGHAVAN, V. V. AND YU, C. T. 1979. Experiments on the determination of the relationships between terms. *ACM Trans. Datab. Syst. 4*, 2, 240–260.

ROBERTSON, S. AND JONES, K. S. 1976. Relevance weighting of search terms. *J. Amer. Soc. Inf. Sci. 27*, 129–146.

ROBERTSON, S. AND WALKER, S. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th ACM SIGIR Conference on Research and Development in Information Retrieval* (Dublin, Ireland). Springer-Verlag, New York, 232–241.

ROBERTSON, S. E., WALKER, S., JONES, S., HANCOCK-BEAULIEU, M. M., AND GATFORD, M. 1995. Okapi at trec-3. In *Proceedings of the Third Text REtrieval Conference (TREC-3)*. E. M. Voorhees and

D. K. Harman, Eds. Department of Commerce, National Institute of Standards and Technology, Gaithersburg, MD, 109–126.

SALTON, G. 1971. *The SMART Retrieval System—Experiments in Automatic Document Processing*. Prentice Hall, Inc., Englewood Cliffs, NJ.

SALTON, G. AND BUCKLEY, C. 1988. Term-weighting approaches in automatic retrieval. *Inf. Proc. Manage. 24*, 5, 513–523.

SALTON, G., BUCKLEY, C., AND YU, C. T. 1982. An evaluation of term dependencies models in information retrieval. In *Proceedings of the 5th ACM-SIGIR Conference on Research and Development in Information Retrieval* (Berlin, Germany). ACM, New York, 151–173.

SALTON, G. AND LESK, M. E. 1968. Computer evaluation of indexing and text processing. *J. ACM 15*, 1 (Jan.), 8–36.

SALTON, G. AND MCGILL, M. J. 1983. *Introduction to Modern Information Retrieval*, 1st ed. McGraw-Hill, New York.

SALTON, G. AND YANG, C. S. 1973. On the specification of term values in automatic indexing. *J. Document. 29*, 351–372.

SILVA, A., VELOSO, E., GOLGHER, P., RIBEIRO-NETO, B., LAENDER, A., AND ZIVIANI, N. 1999. CobWeb—A crawler for the brazilian web. In *Proceedings of the 6th String Processing and Information Retrieval Symposium* (Cancun, Mexico). IEEE Computer Society, Los Alamitos, CA, 184–191.

SONG, F. AND CROFT, W. B. 1999. A general language model for information retrieval. In *Proceedings of the 8th International Conference on Information and Knowledge Management* (Kansas City, MO). ACM, New York, 316–321.

SPINK, A., JANSEN, B. J., WOLFRAM, D., AND SARACEVIC, T. 2002. From e-sex to e-commerce: Web search changes. *IEEE Comput. 35*, 3 (Apr.), 107–109.

SRIKANTH, M. AND SRIHARI, R. 2002. Biterm language models for document retrieval. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Tampere, Finland). ACM, New York, 425–426.

TURTLE, H. AND CROFT, W. B. 1990. Inference networks for document retrieval. In *Proceedings of the 13th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Brussels, Belgium). ACM, New York, 1–24.

VAN RIJSBERGEN, C. J. 1977. A theoretical basis for the use of co-occurrence data in information retrieval. *J. Document. 33*, 106–119.

VAN RIJSBERGEN, C. J. 1979. *Information Retrieval*, 2nd ed. ButterWorths, London, UK.

VOORHEES, E. AND HARMAN, D. 1999. Overview of the Eighth Text Retrieval Conference (TREC 8). In *Proceedings of the 8th Text REtrieval Conference (TREC-8)*. E. M. Voorhees and D. K. Harman, Eds. Department of Commerce, National Institute of Standards and Technology, Gaithersburg, MD, 1–23.

WITTEN, I. H., MOFFAT, A., AND BELL, T. C. 1999. *Managing Gigabytes: Compressing and Indexing Documents and Images*, 2nd ed. Morgan-Kaufmann, San Francisco, CA.

WONG, S. K. M., ZIARKO, W., RAGHAVAN, V. V., AND WONG, P. C. N. 1987. On modeling of information retrieval concepts in vector spaces. *ACM Trans. Datab. Syst. 12*, 2 (June), 299–321.

WONG, S. K. M., ZIARKO, W., AND WONG, P. C. N. 1985. Generalized vector space model in information retrieval. In *Proceedings of the 8th ACM-SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, 18–25.

YU, C. T. AND SALTON, G. 1976. Precision weighting—An effective automatic indexing method. *J. ACM 23*, 1 (Jan.), 76–88.

ZAKI, M. J. 2000. Generating non-redundant association rules. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Boston, MA). ACM, New York, 34–43.

ZOBEL, J. 1998. How reliable are the results of large-scale information retrieval experiments? In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Melbourne, Australia). ACM, New York, 307–314.

ZOBEL, J., MOFFAT, A., WILKINSON, R., AND SACKS-DAVIS, R. 1995. Efficient retrieval of partial documents. *Inf. Proc. Manage. 31*, 3, 361–377.

Received October 2004; revised May 2005; accepted May 2005