A Context Vector Model for Information Retrieval

Holger Billhardt (corresponding author):

Departamento de Inteligencia Artificial, Facultad de Informática, Universidad Politécnica de Madrid. Campus de Montegancedo, 28660 Boadilla del Monte (Madrid), Spain, holger@infomed.dia.fi.upm.es

Daniel Borrajo:

Departamento de Informática, Universidad Carlos III de Madrid. 28911 Leganés (Madrid), Spain, dborrajo@ia.uc3m.es

Victor Maojo:

Departamento de Inteligencia Artificial, Facultad de Informática, Universidad Politécnica de Madrid. Campus de Montegancedo, 28660 Boadilla del Monte (Madrid), Spain, vmaojo@infomed.dia.fi.upm.es

Abstract

In the *vector space model* for information retrieval, term vectors are pair-wise orthogonal, that is, terms are assumed to be independent. It is well known that this assumption is too restrictive. In this article, we present our work on an indexing and retrieval method that, based on the vector space model, incorporates term dependencies and thus obtains semantically richer representations of documents. First, we generate *term context vectors* based on the co-occurrence of terms in the same documents. These vectors are used to calculate context vectors for documents. We present different techniques for estimating the dependencies among terms. We also define term weights that can be employed in the model. Experimental results on four text collections (MED, CRANFIELD, CISI and CACM) show that the incorporation of term dependencies in the retrieval process performs statistically significantly better than the classical vector space model with IDF weights. We also show that the degree of semantic matching versus direct word matching that performs best varies on the four collections. We conclude that the model performs well for certain types of queries and, generally, for information tasks with high recall requirements. Therefore, we propose the use of the context vector model in combination with other, direct word-matching methods.

1 Introduction

In traditional keyword-based IR systems, the documents of a collection are represented by a set of keywords that describe their content. These representations are matched to the words describing the user's information need. Such systems have two fundamental problems: i) the queries have to be specified using the same set of keywords that has been used during document indexing, and ii) the keywords are usually assigned manually to the documents.

More modern IR systems find a way around the problem of a restricted search vocabulary and the subjectiveness of the indexing process by automating this process. There, the representation of a document is based on the words that occur in the text or in some surrogate (e.g., abstract). The set of keywords, or, as it is commonly known in full text indexing methods, the set of *index terms*, includes all the words occurring in the collection. This set is usually confined to only the significant words by eliminating common functional words (also called *stop words*). However, this indexing approach has brought new problems. The first problem is known in the IR community as the *vocabulary* or *word-matching problem*. It refers to the fact that different documents describing the same subject may use different words. In such cases, a simple word-matching approach will probably miss some relevant documents, just because they do not contain the same terms as used in the query. The second problem is the growing need for mechanisms that rank documents in order of relevance, since many more documents are likely to match the words occurring in a query.

Probably the best known model in IR is the *Vector Space Model* (VSM) (Salton, 1989; Salton & McGill, 1983). It implements full-text automatic indexing and relevance ranking. In VSM, documents and queries are modeled as elements of a vector space. This vector space is generated by a set of basis vectors that correspond to the index terms. Each document can be represented as a linear combination of these term vectors. The indexing process thus consists

of calculating the document vectors. During the retrieval process, a query is also put through the indexing process and a query vector is obtained. This query vector is then matched against each document vector and a *retrieval status value* (e.g. *cosine coefficient*) is calculated that measures the *similarity* or *aboutness* of the document to the query. As a result the retrieval process returns a list of all documents ordered by the calculated retrieval status values (relevance to the query).

The effectiveness of retrieval models is usually measured in terms of precision and *recall*. Precision is the ratio of the number of relevant retrieved documents to the total number of retrieved documents. Recall is the proportion of relevant documents retrieved. In Boolean IR systems, where documents are directly retrieved if they satisfy the logical constraints of the query, precision and recall can be calculated directly. In ranking models such as VSM, the calculation of these values is not straightforward since a list of all documents is retrieved. There, common evaluation methods are i) precision at different recall levels (e.g. after 10%, 20%, …, 100% of the relevant documents have been retrieved), ii) precision after a fixed number of retrieved documents, and iii) average precision after all relevant documents, where the precision values are calculated up to each relevant document and these values are averaged.

One assumption of the classical vector space model is that term vectors are pair-wise orthogonal. This means that when calculating the relevance of a document to a query, only those terms are considered that occur in both, the document and the query. Using the cosine coefficient, for example, the retrieval status value for a document/query pair is only determined by the terms the query and the document have in common, but not by query terms that are semantically similar to document terms or vice versa. In this sense the model assumes that terms are independent of each other; there exists no relationship among different terms. It

is well known that this assumption is too restrictive, since words are not actually independent. They do have relationships, because they represent concepts or objects that may be similar or related by many different types of semantic associations. Although the vector space model does not include term dependencies and does not solve the word-matching problem, it has turned out to be very effective, and many other more complex models have not achieved the expected substantial improvement in retrieval performance.

In (Billhardt, Borrajo & Maojo, 2000) we presented our initial ideas on a model that is based on the vector space model but relaxes the independence assumption. This work is further investigated in this article. We use term co-occurrence data to estimate term dependencies. In this way each term can be represented as a vector where the elements correspond to the relationships of the term with other terms. Therefore, each index term can be considered from two points of view: i) as the name of a word in the same sense as it is used in VSM, and ii) as the *semantic meaning* of a word or its *context* in relation to other words. The *term context vectors* are used as the basis for the calculation of semantically richer document representations.

In section 2, we analyze the possibility of introducing term dependencies in the retrieval process. Section 3 describes the model we use in more detail and discusses the different parameters that play a role in the indexing process. In section 4, the retrieval process is explained. There, we also discuss various term-weighting techniques that can be employed. Section 5 shows some experimental results that have been obtained on four different test collections (CACM, CISI, MED and CRANFIELD). In section 6 the effectiveness of the model, measured in terms of precision and recall, is discussed and compared to the vector space model using *Inverse Document Frequency* weights (IDF weights). Finally, section 7 gives some conclusions and directions for future research.

2 Term Dependencies in Document Indexes

One of the fundamental problems in information retrieval is the vocabulary or word-matching problem. It actually refers to two different things: i) the same objects may be expressed in different ways so documents about the same issue may use different words, and ii) the existence of words that have different meanings. The first is called *synonymy* and the second *polysemy*. In ranking IR systems, the prevalence of synonyms tends to decrease precision at higher recall levels (e.g. the last relevant documents found are further down in the list), since not all of the relevant documents may match the terms in a query. On the other hand, the existence of polysemic terms is related to low precision at low recall levels, that is, there will be less relevant documents at the beginning of the list. If a term with an unclear meaning is used in a query then many irrelevant documents, which contain the term but not with the intended meaning, are likely to be retrieved earlier. Polysemy and synonymy actually describe just the extreme cases and there is a broad spectrum of relationships between words and their meanings between the two.

There have been many attempts to solve the vocabulary problem. It has been argued that the use of term dependencies can help to achieve this aim (Bollmann-Sdorra & Raghavan, 1998; Raghavan & Wong, 1986) and almost all methods use such term dependencies in one way or another. Most of these methods get term relationships from co-occurrence data, that is, from the frequency terms co-occur in the same documents. Schütze (1992) represents the semantics of words and contexts in a text as vectors in a vector space where the dimensions correspond to words. These vectors are called *context vectors*. A context vector for an entity is obtained from the words occurring close to that entity in a text. Therefore, the vectors represent the context of a single occurrence of a word. Because of the high dimensionality of the vectors, dimensionality reduction is carried out by means of

singular value decomposition. The vectors are later applied to word sense disambiguation and thesaurus generation tasks. Even though the use of the methods is only reported for these tasks, they could be used directly in IR systems as well.

Many of the methods approach the vocabulary problem by means of *query expansion*. Query expansion techniques either consider the information search as a process and try to refine a user's query in each retrieval step (Chen et al., 1997; Chen et al. 1995) or they use inter-term relationships to expand a query when it is received. The information on term dependencies may come from manually or automatically generated thesauruses, as in (Gotlieb & Kumar, 1968; Chen et al., 1997; Chen et al. 1995; Jing & Tzoukermann, 1999), or may be obtained by means of relevance feedback. In relevance feedback, the system analyses the documents a user judged relevant at a previous stage and uses this information for query refinement. It has been shown that automatic query expansion using relevance feedback can add useful words to a query and can improve retrieval performance (Salton & Buckley, 1990). However, such techniques require the collaboration of the user who has to judge the documents supplied and such judgments are often not provided. *Ad hoc* or *blind* feedback can solve this problem (Buckley et al., 1995; Mitra, Singhal, & Buckley, 1998). Blind feedback assumes that the top retrieved documents of a retrieval process are likely to be relevant and that shifting the query towards these documents will improve the retrieval results. In a blind feedback approach, documents are first retrieved for the original query. Then the top n documents are selected and used in a relevance feedback process to create an expanded query. Usually those terms that are good representatives of the selected documents are added to the query. Afterwards, the final document ranking is obtained by repeating the retrieval process with the new, expanded query. Besides the effect of shifting the query towards a region of possibly relevant documents in the vector space, blind feedback expands the query by adding

more terms to it. It is likely that new terms are added, which are semantically related to the original query terms. Thus, probably more relevant documents will match the terms in the new query. On the other hand, nonrelevant documents that matched some terms in the original query, but use these terms in a different meaning, may now get a lower retrieval status value since the new query terms may not be present. Blind feedback can improve the retrieval performance considerably if the selected documents are actually relevant. If this is not the case, however, the performance may also decrease.

Another class of methods approach the vocabulary problem by generating representations of documents and queries that are semantically richer then just vectors based on the occurrence frequency of terms. They use the *inherent semantic structure* that exists in the association of terms with documents in the indexing process. In *Latent Semantic Indexing* (LSI) (Deerwester et al., 1990), singular value decomposition is used to decompose the original term/document matrix into its orthogonal factors. Of those, only the *n* highest factors are kept and all others are set to 0. The chosen factors can approximate the original matrix by linear combination. Thus, smaller and less important influences are eliminated from the document index vectors, and terms that did not actually appear in a document may now be represented in its vector. LSI can be seen as a space transformation approach, where the original vector space is transformed to some factor space. The *Probabilistic Latent Semantic Indexing* (PLSI) by Hofmann (1999) has a similar approach. Also, in PLSI the documents are mapped to a reduced vector space, the *latent semantic space*. As opposed to LSI, the model has a solid statistical foundation. It is based on a latent variable model for general co-occurrence data, which associates an unobserved latent class variable with each observation. The number of latent factors will be much smaller than the number of words and the factors act as prediction variables for words. The factors are obtained using a generalization of the

Expectation Maximization algorithm. A transformation of the vector space is also the basis of the *Generalized Vector Space Model* (GVSM) (Wong et al., 1987). Wong et al. argue that the orthogonality assumption in VSM is too restrictive and, therefore, another representation has to be found. Analyzing term correlations obtained from co-occurrence data, they define a new set of orthogonal basis vectors that spans a (transformed) vector space. This space is then used to represent document and query vectors more semantically, since term dependencies are implicitly included. The results they report with their model show a clear improvement over the classical vector space approach.

The work presented in this paper is similar to the *Distributional Semantics based Information Retrieval* (DSIR) method proposed by Rungsawang and Rajman (Rungsawang, 1999; Rungsawang & Rajman, 1995). They also use term co-occurrence information, collected over the whole document collection to get term vectors in the same space as document and query vectors. Then, vectors representing the documents in the system are obtained using the term vectors of the words occurring in those documents. Even though in its basic aspects their method is quite similar to ours, we used a different indexing and term weighting approach, which leads to better results.

3 Document Indexing Based on Term Contexts

The hypothesis of our approach is that users of IR systems have an understanding of possible documents they are looking for which may not correspond to the actual representation of the documents. In fact a user normally specifies a query with respect to his/her particular understanding of documents. This understanding, on the other hand, is determined by his/her knowledge of the real world, obtained, in part, by reading real documents.

In our work we try to find a semantic representation of documents, which estimates the

way hypothetical users could understand the documents. However, we do not consider particular users. Rather we look at all documents in a collection and try to automatically understand each document based on the information of the whole collection. This is somehow similar to how a user may understand a document by knowing all the documents of a collection.

We assume that document understanding can be represented by a set of weighted semantic factors, which we call *document context vectors*. As factors we choose the semantic concepts underlying the index terms. Thus, each element in the representation has a corresponding element in a traditional document description based on term frequency vectors. The basic difference of our representation to the standard term frequency vectors is that the values in document context vectors are not only determined by the occurrence frequency of the corresponding term itself, but also by all the other terms occurring in the document. That means, we assume that each term in a document indicates the possibility of existence of concepts in that document, which correspond to other terms. In other words, each term has a certain influence on the existence of every other term. We use term context vectors to represent the relations or influences between terms.

In the following we give a formal description of the indexing process. The basic algorithm that we use consists of three steps: i) compute term/document matrix, ii) generate term context vectors, and iii) transform document vectors based on occurrence frequencies to document context vectors.

Normally, the starting point of the indexing process in a vector space based method is a term/document matrix as shown below:

	d_1	d_2	…	d_m
t_1	w_{11}	w_{21}	…	w_{m1}
t_2	w_{12}	w_{22}	…	w_{m2}

$$\begin{array}{c|cccc} \vdots & \vdots & \vdots & \ddots & \vdots \\ t_n & w_{1n} & w_{2n} & \cdots & w_{mn} \end{array}$$

m is the number of documents in the collection and *n* the number of index terms. Each element w_{ij} in this matrix corresponds to the occurrence frequency of term t_j in document d_i. The vectors $\vec{d}_i = (w_{i1}, w_{i2}, \ldots, w_{in})^T$ are used as the initial document vectors in most information retrieval models that are based on the *bag of words* approach. In the classical VSM, these vectors represent the documents of a collection. They are usually only modified for normalization purposes and through the introduction of term weights to improve retrieval performance (e.g., IDF weights). However, in our approach we use the initial occurrence frequency vectors as the starting point for calculating document context vectors.

3.1 Generating Term Context Vectors

Term context vectors can be seen as a semantic description of terms, which reflect the influences of terms in the conceptual descriptions of other terms. The set of term context vectors can be represented by an $n \times n$ matrix T as follows:

$$T = \begin{pmatrix} c_{11} & c_{21} & \cdots & c_{n1} \\ c_{12} & c_{22} & \cdots & c_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ c_{1n} & c_{2n} & \cdots & c_{nn} \end{pmatrix}$$

where the i-th column represents the term context vector $\vec{t}_i = (c_{i1}, c_{i2}, \ldots, c_{in})^T$ for the index term t_i in the *n* dimensional term space. Each c_{ik} represents the influence of term t_k on term t_i. The use of term context vectors allows us two different interpretations of index terms: i) a syntactic interpretation of a term as the word it stands for, and ii) as a semantic description of a concept underlying the word the term stands for. Term dependencies can be integrated into the indexing model by means of the second interpretation, that is, by

describing each index term by a set of concepts, where each concept has a directly corresponding index term.

In the following, we discuss how the elements of matrix T can be obtained. To estimate the influence of term t_k on the semantic description of t_j we use the co-occurrence frequency of both terms, that is, the frequency with which t_k and t_j co-occur in the same textual units across the whole collection.

We define two different co-occurrence frequency measures:

$$c_{ij} = \begin{cases} 1 & \text{, if } i = j \\ \dfrac{\sum_{k=1}^{m} w_{ki} w_{kj}}{\sum_{k=1}^{m} \left(w_{ki} \sum_{a=1, a \neq i}^{n} w_{ka} \right)} & \text{, otherwise} \end{cases} \quad (1)$$

$$c_{ij} = \begin{cases} 1 & \text{, if } i = j \\ \dfrac{\sum_{k=1, \text{ where } w_{kj} \neq 0}^{m} w_{ki}}{\sum_{k=1}^{m} w_{ki}} & \text{, otherwise} \end{cases} \quad (2)$$

With both definitions, it is assured that for all $i \leq n$ and all $k \leq n$: $c_{ik} \leq 1$. Furthermore, for all diagonal elements c_{ii} it holds $c_{ii}=1$. These properties are quite important from an interpretative perspective. They imply that the influence of a term in its context vector is bigger than the influence of any other term on that term. As we see later, however, it may be advantageous to set these influences to 0. Then, only its relations to all other terms describe the semantic meaning of a term.

In equation (1) the influence term t_j has on term t_i is defined in a probabilistic manner. If we pick up a word t_i from document d_k, then c_{ij} is the probability of picking up the word t_j

from the same document d_k. Thus, $c_{ij} = P(t_j/t_i)$. Equation (2) is a rather intuitive definition of the elements of a term context vector. Compared to equation (1) the non-diagonal elements have higher values. Therefore, the importance of term t_i in its context vector is relatively lower. Furthermore, the difference between both equations can be seen if we look at the following situation. A term t_i occurs one time in a document where another term t_j occurs quite often. Then, if we just consider this document, the concept factor c_{ij}, that is, the influence of t_j in the meaning of t_i would be relatively high since somehow all occurrences of t_j count. Moreover, if t_i occurs just in some more documents where t_j does not occur, then the calculated influence of t_j on t_i would certainly not correspond to the "real" influence. On the other hand, using equation (2), this would not happen. Therefore, equation (2) avoids undesired situations of this kind.

In both equations we used co-occurrence in the sense of terms that occur together in the same document. However, one could also define co-occurrence on basis of textual units other than entire documents. Phrases or paragraphs may be used or a window may be defined that is "shifted" over the texts and the co-occurrence of terms within the same window can be counted. The latter approach is slightly more complicated, because it is necessary to assure that the same pairs of terms are not counted twice. This may happen since the window "slides" word-wise over the text and, therefore, the same pair of terms may be part of two or more windows. We also did some experiments with the "sliding window" approach, but they performed worse and are omitted in this article.

3.2 Generating Document Context Vectors

Once the term correlation matrix T has been generated, we transform each initial document vector $\vec{d}_i = (w_{i1}, w_{i2}, ..., w_{in})^T$ into a context vector $\vec{d}'_i = (c_{i1}, c_{i2}, ..., c_{in})^T$. This is

done with the following equation:

$$\vec{d}'_i = \frac{\sum_{j=1}^{n} w_{ij} \frac{\vec{t}_j}{|\vec{t}_j|}}{\sum_{j=1}^{n} w_{ij}} \quad (3)$$

where $\vec{t}_j$ is the context vector of term t_j and $|\vec{t}_j|$ is the length of vector $\vec{t}_j$ (we use the Euclidean vector norm to define the length of a vector). The division of the elements in term context vectors by the length of the vector is just a normalization step. However, this normalization may be omitted since the vectors are not biased by the collection occurrence frequencies of the terms. In fact, in our experiments we did not see a significant change in the experimental results using one approach or the other.

The generated document context vectors $\vec{d}'_i = (c_{i1}, c_{i2}, \ldots, c_{in})^T$ correspond to the centroid of the term context vectors of all terms belonging to the document. Thus, the value of concept c_k in the document context vector for document *i* is the average of the influences of term t_k on all terms occurring in d_i. Also, if matrix T is the identity matrix (e.g. all terms are represented only by themselves), then the context vectors for the index terms are pair-wise orthogonal and the resulting vectors have the same direction than the normal term frequency vectors. Only the scaling is different. This means that in this case no term dependencies are considered and our model behaves in the same way as the classical vector space model. Considering this fact it is interesting to analyze different ways of using the term context vectors. If we set all non-diagonal elements in T to 0, then the model behaves like VSM. On the other hand, if we set the diagonal to 0, then we get the other extreme. That is, the influence of a term in a document is only determined by all other terms that also occur in the document. Leaving all elements in term context vectors as defined above gives an intermediate model, but equation (2) gives

more importance to the non-diagonal elements, thus the "other" terms, than equation (1). In the experiments we tried both equations with and without setting the diagonal elements in T to 0.

4 Document Retrieval

Once the indexing process has been completed and all document context vectors are stored in the system, it can receive queries. For each query, it computes a list that presents documents in order of relevance. To calculate the relevance of a document to a query we use the standard cosine similarity measure:

$$s(d_i, q) = s(\vec{d'}_i, \vec{q}) = \frac{\sum_{j=1}^{n} p_q(j) q_j \cdot p_d(j) c_{ij}}{\sqrt{\sum_{j=1}^{n} (p_q(j) q_j)^2} \sqrt{\sum_{j=1}^{n} (p_d(j) c_{ij})^2}} \quad (4)$$

where $\vec{d'}_i = (c_{i1}, c_{i2}, \ldots, c_{in})^T$ is the document context vector for document d_i and $\vec{q} = (q_1, q_2, \ldots, q_n)^T$ is a vector representing the query. Furthermore, $p_q(j)$ and $p_d(j)$ are some term weights applied to the components of the query and documents vectors, respectively. Documents are presented in order of decreasing cosine similarity.

Query vectors may be computed in different ways. In our experiment we used three different approaches: i) simple term frequency vectors for representing queries, ii) binary query vectors that indicate the existence of terms in the query (the frequency of the terms is omitted), and iii) query context vectors obtained in the same way as document context vectors. The first approach is possible since the concept space is spanned over all index terms. From an interpretative point of view, when applying frequency vectors the term frequencies of the terms in the query are considered as values for the corresponding concepts in the concept space. In the second approach each query term represents just the existence of the

corresponding concept in the query but all the concepts are considered equally important. This may be appropriate in some queries where words that occur more often do not have a high semantic value. There, the importance of such words is “filtered out”. In our experiments we found out that our model performs better on some collections, when binary query vectors are used. This was also experienced by Wong et al. (1987). The third query indexing approach transforms queries in the same way as documents and, thus, increases the number of concepts that represent a query. In our experiments this method works better than the other approaches in some cases. However, considering memory, storage and calculation time requirements, the use of query context vectors is more costly than the other methods. Using vectors based on simple term frequencies drastically reduces retrieval time, because query indexing is much shorter and, more importantly, because the query vectors are sparse. Depending on the number of index terms, a query context vector will possibly have several thousands non-zero elements, whereas a term frequency or binary vector will only have a couple of non-zero values.

Equation (4) includes a length normalization of the elements in the query and document vectors. (As pointed out earlier, we use the Euclidean vector norm to describe the length of a vector.) It is well known that differences in the length of document vectors may lead to worse retrieval results. Using only the scalar product as a similarity measure, for example, would mean that longer documents would have a higher probability of matching terms in a query than shorter ones. Therefore, they will be ranked higher. This also applies to our model, albeit for a slightly different reason. The document context vectors are already quite similar in length. Actually, if the normalization of term context vectors is used in equation (3), their length is always less than or equal to 1. This is because the term context vectors used in the indexing process are normalized to a length of 1. In fact, the closer the

meaning of the terms occurring in a document, the closer the length of its vector will be to one. Even though this is an apparently interesting property we did not study it in more detail. Another variation in the length is caused by the multiplication with term weights. The normalization of query vectors has no impact on ranking and, therefore, may be omitted. This is because for a given query the same normalization applies to all documents. Thus, equation (4) may be simplified to:

$$s(d_i, q) = s(\vec{d'}_i, \vec{q}) = \frac{\sum_{j=1}^{n} p_q(j) q_j \cdot p_d(j) c_{ij}}{\sqrt{\sum_{j=1}^{n} \left(p_d(j) c_{ij}\right)^2}} \qquad (5)$$

The only advantage of using (5) instead of (4) is the reduction in calculation time. This can be quite important in on-line retrieval systems, where the response time is an issue of concern.

4.1 Term Weights

It is well known in the IR community, that the use of appropriate term weights can considerably improve the retrieval performance of a system. In the vector space model, each element of the original document vectors, that is, of the columns in the document/term matrix can be multiplied by such weights. Many different term-weighting schemata have been proposed, e.g., the signal weight (Dennis, 1967) or the discrimination value (Salton & Yang, 1973). However, *Inverse Document Frequency* (IDF) is probably the most commonly used and also one of the most effective weighting schemas. The IDF weight of a term t_i is calculated as follows:

$$idf(t_i) = \log_2\left(\frac{m}{df(t_i)}\right) + 1 \qquad (6)$$

where *m* is the number of documents in the collection and $df(t_i)$ is the document frequency of term t_i, that is, the number of documents, in which the term occurs. The reason for using IDF weights is that terms that occur only in very few documents are better discriminators than terms that occur in the majority of the documents and, therefore, should be weighted higher.

Term-weighting techniques can also be expected to improve retrieval performance in the context vector model. Indeed, this is the case, as the presented results will show. Taking advantage of the fact that concepts and terms have a direct correspondence in our model, we can use the standard *idf* weight as defined in equation (6). Moreover, the difference of the context vector model to the classical VSM allows us to define other term weights, which will be described below.

Deviation measures of terms across document vectors.

Even though the classical IDF weights consider the number of different documents in which a term occurs, they do not consider the differences in the number of times a term occurs in those documents. As argued for IDF weights, it seems obvious that the variations in the occurrence frequency of terms in the documents will also be important. For example, a term that occurs in all documents with very different frequency values in each one will be more important than a term occurring in all documents just once. The IDF values in both cases, however, will be 1. This leads to the conclusion that some kind of deviation measure that assesses the differences of the occurrence frequency of terms across all documents should improve retrieval performance.

The motivation for using weights based on deviation values can also be seen from another point of view. As compared to the document vectors from the original term/document matrix, the document context vectors calculated in (3) are not very sparse. This is because

they are calculated using not only those terms that actually occur in the documents, but also other terms co-occurring with them in at least one textual unit in the collection. Therefore, the number of non-zero elements in a document context vector will be much higher than the number of different index terms in the document. Furthermore, the values of the elements of document context vectors are real and not natural numbers. Because of these two properties there is no direct correspondence of *idf* weights in the context vector model. However, other measures that describe the deviations of concepts across the documents seem to represent a similar idea and can be easily computed.

Let *m* be the number of documents in the collection and *n* the number of index terms. Let $\vec{d}'_i = (c_{i1}, c_{i2}, \ldots, c_{in})^T$ (for $1 \leq i \leq m$) be the context vectors of the documents. We defined the following measures to assess the deviation of concept values across all document context vectors:

- Modified mean average deviation:

$$\text{dcvmamd}(t_j) = 1 + \frac{\sum_{i=1}^{m} \left| \frac{\text{norm}(c_{ij})}{\text{norm\_mean}(c_j)} - 1 \right|}{m} \tag{7}$$

- Modified variance:

$$\text{dcvmvar}(t_j) = 1 + \log_2 \left(1 + \frac{\sum_{i=1}^{m} \left(\frac{\text{norm}(c_{ij})}{\text{norm\_mean}(c_j)} - 1 \right)^2}{m - 1} \right) \tag{8}$$

- Combination of *idf* and *dcvmamd*:

$$\text{idfdcvmamd}(t_j) = 1 + \text{idf}(t_j) \frac{\sum_{i=1}^{m} \left| \frac{\text{norm}(c_{ij})}{\text{norm\_mean}(c_j)} - 1 \right|}{m} \tag{9}$$

- Combination of *idf* and *dcvmvar*:

$$\text{idfdcvmvar}(t_j) = 1 + \text{idf}(t_j) \cdot \log_2 \left(1 + \frac{\sum_{i=1}^{m} \left(\frac{\text{norm}(c_{ij})}{\text{norm\_mean}(c_j)} - 1 \right)^2}{m - 1} \right) \qquad (10)$$

$$\text{where } \text{norm}(c_{ij}) = \frac{c_{ij}}{\sqrt{\sum_{k=1}^{n} c_{ik}^2}} \text{ and } \text{norm\_mean}(c_j) = \frac{1}{m} \sum_{i=1}^{m} \frac{c_{ij}}{\sqrt{\sum_{k=1}^{n} c_{ik}^2}}.$$

$\text{norm}(c_{ij})$ is the j-th component in the document context vector $\vec{d}'_i$, when this vector is normalized to the Euclidean length of 1, and $\text{norm\_mean}(c_j)$ is the mean of this value across all document context vectors.

Equation (7) is a modification of the average mean deviation and equation (8) a modification of the variance. The modification we used is a normalization step that scales all concept values such that their mean is 1. Without this normalization, concepts with higher values in the average will usually get a higher weight. The normalization step eliminates the influence of the mean in the weights. In equation (8) we also took the logarithm of the variance to scale the higher values down. Both equations measure the deviation in the distribution of concepts across all documents. They give higher weights to those terms whose corresponding concepts have very different values in all documents. On the other hand, a concept that occurs quite equally in all document vectors, and, thus, is a bad semantic discriminator leads to a lower concept weight.

idfdcvmamd and *idfdcvmvar* are just combinations of these weights with the standard *idf* weights as defined in equation (6). Finally, we add one to the equations in order to generate weights that are greater than 1.

We used the same equations to compute the corresponding deviation measures of term occurrence frequencies across the initial document vectors $\vec{d}_i = (w_{i1}, w_{i2}, ..., w_{in})^T$ (for $1 \le i \le m$). These weights are denoted by *dtfmamd*, *dtfmvar*, *idfdtfmamd*, and *idfdtfmvar*.

Deviation measures of concepts in term context vectors.

Another weighting technique is based on the deviation of the elements in the term context vectors. These weights are only indirectly based on the document collection. They can be calculated using the term context vectors in the matrix T; the original term/document matrix is not used.

Let *n* be the number of index terms and let $\vec{t}_i = (c_{i1}, c_{i2}, ... , c_{in})^T$ (for $1 \le i \le n$) be the context vectors of terms t_i. We define similar measures as in the case of the deviation of concept values across document vectors:

- Modified mean average deviation of term context vectors:

$$\text{tcvmamd}(t_i) = 1 + \frac{\sum_{j=1}^{n} \left| \frac{c_{ij}}{\text{mean}_{\vec{t}_i}} - 1 \right|}{n} \qquad (11)$$

- Modified variance of term context vectors: $\text{tcvmvar}(t_i) = 1 + \frac{\sum_{j=1}^{n} \left(\frac{c_{ij}}{\text{mean}_{\vec{t}_i}} - 1 \right)^2}{n-1}$ (12)

- Combination of idf and *tcvmamd*: $\text{idtcvfmamd}(t_i) = 1 + \text{idf}(t_i) \frac{\sum_{j=1}^{n} \left| \frac{c_{ij}}{\text{mean}_{\vec{t}_i}} - 1 \right|}{n}$ (13)

- Combination of *idf* and *tcvmvar*: $\text{idftcvmvar}(t_i) = 1 + \text{idf}(t_i) \frac{\sum_{j=1}^{n} \left(\frac{c_{ij}}{\text{mean}_{\vec{t}_i}} - 1 \right)^2}{n-1}$ (14)

where $\text{mean}_{\vec{t}_i}$ is the mean of all values in the term context vector of term t_i.

The motivation for using *tcvmamd* and *tcvmvar* weights is as follows. A term that is related to all other terms more or less in the same way is not very descriptive; it does not have a very specific meaning. Such terms should have less weight in the representation of documents. In fact, common function words, like “the”, if they were not filtered out in indexing, would fall into this category since they will have a very similar relationship to all other terms. Considering the term context vectors of such terms, they will be very uniform and the weighting techniques defined above would give them little weight. On the other hand, terms that are strongly related to only a few other terms represent more specific meanings and should get higher weights. We used the same normalization as in equations (7) to (10) to eliminate the influence of the mean on the deviation measures and, thus, on the term weights.

It is also interesting to point out that the weights defined here can be easily calculated once the term context vectors are defined; and that they do not directly depend on the documents in the collection. If we consider that term relationships may be obtained from other sources than the documents of a collection, then the specified weights do not change when the collection is changed.

5 Experimental Results

In this section, we present experimental results on four different test collections, which have been extensively used for evaluating IR systems. The collection characteristics are as follows:

- MED is a collection that comprises 1033 medical abstracts. The collection has 30 queries.
- CRANFIELD is a collection of 1398 documents from aerodynamics and 225 associated queries.

- CISI contains 1460 documents and 76 queries from library science.
- CACM is a collection of 3204 documents and 52 queries.

In the indexing process some pre-processing was done before context vectors had been calculated for documents and queries. First, we eliminated common function words (stop words) by using the stop word list that comes with the SMART system at *ftp://ftp.cs.cornell.edu/pub/smart/*. Afterwards, the word stem of each remaining word was calculated using the Porter stemmer algorithm (Porter, 1980). We then eliminated all those word stems that occurred just once in the collection. The remaining word stems build the set of index terms t_i with $0 \leq i \leq n$ that was used to calculate context vectors.

With this base line, we tested the use of binary, term frequency and context query vectors, and three different types of term context vectors (using equation (1) and (2) for calculating term context vectors and also using equation (1) with setting the diagonal elements in the matrix to 0). We also tested equation (2) with elimination of the diagonal elements in the matrix, but no improvements were obtained, so the results are not reported in this paper. Each of the combinations has been tested for different weighting schemata. We tried 14 weighting techniques and their combinations with respect to query and document term weights. Besides the weights described in section 4.1, we also specified a *no* weight which is just 1 for each term. With this parameterization, each test run is denoted by *mmmm_qqqq_dwdw_qwqw*, where

- *mmmm* specifies the calculation of the term correlation matrix T, and, thus, the term context vectors: *probdiag* denotes the probabilistic definition from equation (1), *probnodiag* means the use of equation (1) but with elimination of the diagonal elements (diagonal set to 0), and *intudiag* specifies the use of the intuitive definition (equation 2)
- *qqqq* specifies the type of query vector used: *bin* stands for binary, *tf* for term frequency,

and *qcv* for query context vectors, respectively

- *dwdw* and *qwqw* denote the document and query term weights applied and may be one of the following: *no* (a weight of 1 for all terms), *idf* (inverse document frequency), *dcvmamd* (modified average mean deviation of concepts values across documents), *dcvmvar* (modified variance of concept values across documents), *dtfmamd* (modified average mean deviation of term occurrence frequencies across documents), *dtfmvar* (modified variance of term occurrence frequencies across documents), *tcvmamd* (modified average mean deviation of concept values in term context vectors), *tcvmvar* (modified variance of concept values in term context vectors), and *idfdcvmamd*, *idfdcvmvar*, *idfdtfmamd*, *idfdtfmvar*, *idftcvmamd* and *idftcvmvar* as the combination of the different weights with *idf*.

The results of the test runs were compared to standard VSM using IDF weights (denoted as *vsm*). We also give results for a blind feedback approach with *vsm* (denoted *vsm_bf*) There, we applied the same feedback method as used by Buckley et al. (1995) using Rocchio's formula (Rocchio, 1971):

$$q_{new} = \alpha \cdot q_{old} + \beta \cdot \text{average\_wt\_rel\_docs} + \gamma \cdot \text{average\_wt\_nonrel\_docs} \quad (15)$$

In the initial run (with *vsm*) the top five documents were assumed to be relevant and were used for query expansion. The queries were expanded by up to 300 terms, where we chose those terms that occurred in more of the selected documents. The Rocchio parameters were set to $\alpha = \beta = 8$, and $\gamma = 0$.

In the *vsm* and *vsm_bf* runs we indexed queries in the same way as documents, that is, we did not use binary query vectors. The results for *vsm* and *vsm_bf* have been obtained using our system and employing the same set of index terms as for all the other methods. Also, the evaluation methods used are the same as for all other runs. This ensures greater consistency in

the comparison.

In Table 1 we present the average precision over all relevant documents for the best weight combination and the different query vector and term context vector strategies. The average precision is calculated for each query and further averaged over all queries in a collection. For this measure, we used the same calculation method as it is used in TREC (Vorhees & Harman, 1999). We also give the t-statistics for the runs that show improvements over *vsm*. These values have been calculated using a t-test for paired observations. A value of about 1.7 or higher indicates a statistically significant improvement at a significance level of 5% on each of the four collections.

6 Discussion

It can be easily seen that the context vector approach performs quite differently on the four collections. It works very well on MED, and has less advantage on the other collections. Considering the different matrix specifications, one can argue that there is a certain ordering of the models with respect to the importance that is given to a term in its own context vector, that is, how much a term should represent its own concept. If the term correlation matrix is the identity matrix, the standard VSM is simulated; in this case each concept is only described by the corresponding term. On the other hand, if the diagonal elements in the term correlation matrix T are set to 0, the strongest concept matching is achieved. It should also be considered that the intuitive matrix (equation (2)) gives less importance to each term in its concept (see section 3.1). Then *probdiag* < *intudiag* < *probnodiag* specifies an ordering of the three matrix models where *a* < b means that using model *b* implies stronger concept matching and less direct word matching than using model *a*. In this sense, *probdiag* is the closest model to standard VSM, and thus, word matching, and *probnodiag* implies the strongest concept matching approach. Considering this, Table 1 shows that the degree of concept matching

versus direct word matching that performs best varies on each collection. On MED the strongest semantic matching performs best, CRANFIELD requires an intermediate approach, and CACM and CISI perform best when the use of term relationships is relatively small. However, on all four collections retrieval performance can be reliably improved if term relationships are taken into account.

We believe that the fact that strong concept matching performs well on MED and poorly on the other collections is partially due to the different collection characteristics. In CISI, CACM, and CRANFIELD, the documents are probably more semantically alike than in MED. This means that in these collections the different words are used in more or less the same contexts. In such a situation a strong concept matching will be too general and will not recognise the small differences in the documents that a syntactic word matching would find. In this sense, our approach seems to have advantages when the domain of the collection is rather wide. Considering the problems of synonymy and polysemy, it can be expected that the context vector model will treat them better than VSM. However, in very closed domains these problems do not occur very strongly. Another reason may be the different length of the documents. In CACM and CISI, for example, documents are shorter than they are in MED and this may lead to more arbitrary and biased term context vectors. Therefore, if too much importance is given to the relationships among the different terms, the retrieval results will degrade. Term relationships obtained from larger corpora than just the collection under consideration can be expected to lead to better term context, and, thus, document descriptions. One advantage of our model is that it facilitates such an approach, since, in principle, term dependencies are independent of the particular collection. In this sense, the term correlation matrix can be “learnt” from large corpora as a form of general language understanding. However, one should consider that certain types of term relationships would depend on the

domains of the corpora on which they have been obtained. At least for these relationships, it seems clear that they can only be “learnt” on larger corpora if the domain of these corpora is the same as the domain of the collection under consideration.

Another basic difference in the collections concerns the query characteristics. Queries in MED define rather vague or general information needs. On the other hand, queries in CACM and CISI are very specific in the sense that they ask for very specific information. In such cases, a less semantic matching approach will perform better than strong semantic context vector comparison, since the latter describes the concepts rather vaguely. However, vagueness or generality in user’s information needs may be intended (Efthimiadis, 1993), and there our approach seems to have advantages. In general we observed that the context vector approach performs better on queries with more relevant documents. This seems to confirm that for very precise queries, that is, when only very few relevant documents exist, the standard VSM works better. However, our method performs better for some queries with few relevant documents, probably because the query terms are not directly contained in those documents.

As opposed to MED, quite a few queries in CACM contain many different “significant” words, sometimes specified in an “OR” like manner. This usually implies that the context vector model, when used with term frequency based query vectors, will find too many documents that are semantically similar to the query. Also here, direct word-matching seems to have more power to differentiate between those documents.

Regarding query encoding, it can be seen from the results in Table 1 that in MED and CRANFIELD the use of binary query vectors has advantages. On the other hand, in CACM and CISI query vectors based on term frequencies perform better. The use of binary vectors results in a significant degradation in CISI, whereas on the other collections the difference

between *tf* and *bin* query vectors is not very strong. The differences detected in these aspects are not actually related to the context vector model. They are due to the characteristics of the queries. In CACM and CISI queries are generally longer and more descriptive. In general, it seems that the most significant words occur just once in short queries but more often in long queries. Therefore, in long queries, using term frequency vectors instead of binary vectors will give relatively more importance to the significant terms. However, the less important words also count more in such a case, but their importance can be filtered out by appropriate term weighting techniques. From the point of view of short queries, if the significant terms occur just once and the non-significant terms more times, it seems clear that a binary encoding is more suitable.

With respect to the use of query context vectors it seems that this method works relatively well in combination with the *probdiag* and the *intudiag* matrix. The only exception is CISI in the *intudiag* runs. Especially on CACM the improvement over *tf* or *bin* queries seems important. We do not have a clear explanation for this. It might be due to what we said before, that the number of significant terms is generally greater in CACM queries. As we mentioned above, in such cases the context vector model with *tf* or *bin* queries may not be able to find the most relevant documents out of the many documents that are similar to the query. Considering the concept space, the search is widened since the different concepts corresponding to the significant words have more or less the same importance. If we use query context vectors in these situations, then the widening process may be inverted because only those terms or concepts that are strongly related to most of the significant words will have the strongest weight in the query. However, the use of *qcv* may also lead to an even more general search and, thus, precision at low recall levels may decrease. This might be the reason for the bad performance on CISI.

Regarding the different weighting techniques, the best performing weights when using *tf* or *bin* queries were: *idfdcvmamd_idftcvmamd*, *idfdcvmamd_idfdtfmamd*, *idfdcvmamd_idfdtfmvar*, and *idftcvmamd_idftcvmamd*. These weight combinations behave quite similar on all collections, except in the *intudiag* runs on CACM. There, the best combination is *idftcvmvar_dtfmvar*, which also works well on CRANFIELD but quite worse on MED and CISI. Generally, it seems that *idfdcvmamd_idfdtfmamd* is the most recommended weight combination for *tf* or *bin* queries. In the tests with query context vectors, the weighting schemata behaved differently for *intudiag* and *probdiag*. In the first case, the best weights were: *dcvmamd_idftcvmvar* and *dtfmamd_idftcvmvar*, and in the *probdiag* runs: *dcvmamd_idfdtfmvar*, *dtfmamd_idfdtfmvar*, and *dcvmamd_idfdcvmvar*. Generally the *probdiag* matrix performed better than *intudiag* when *qcv* query vectors are used and the best choice here seems to be this matrix type together with the *dcvmamd_idfdtfmvar* weight combination. In general, statistically significant improvements on all collections (at a significance level of 5%) can be obtained for five *probdiag/qcv* and 16 *probdiag/tf* runs, and the best combination is *probdiag_qcv_dcvmamd_idfdtfmvar* with improvements of 10.4% for CACM, 10.1% for CISI, 3.3% for CRANFIELD, and 12.1% for MED. There is one *intudiag* run (*intudiag_tf_idfdcvmamd_idfdtfmamd*) that performs statistically reliably better than VSM on three collections (it performs worse on CACM). The relative improvements are –4.4% (CACM), 8.1% (CISI), 5.1% (CRANFIELD), and 23.7% (MED). This implies, that with the use of *intudiag* the results are less robust, but higher improvement gains can be obtained.

Compared to the blind feedback approach (*vsm_bf*), the best context vector method performs better on all collections except MED. Moreover, the results obtained with *vsm_bf* are only statistically significant at a significance level of 5% for MED. This is basically due to

high variance of this approach. Blind feedback relies on the fact that the top selected documents are actually relevant, and if this is not the case it will generally perform worse. In fact, only in MED the number of queries where *vsm_bf* performs better than *vsm* is considerably higher than the number of queries where it performs worse.

A comparison of our results to the results reported by Hofmann (1999) for LSI and PLSI seems difficult. In terms of absolute average precision values, the best context vector methods perform better than PLSI and LSI. However, the average precision values he reports for the baseline (*vsm*) are also much lower than the values we calculated. Thus, the improvement gains he reports are higher than ours for the PLSI methods and in the same range for LSI. The differences in the absolute values may be due to the use of different calculation methods and /or different preprocessing techniques. Furthermore, Hofmann used combinations of LSI and PLSI with the standard VSM and does not report the "pure" LSI and PLSI results. It can be expected that an appropriate combination of semantic methods with direct word-matching works generally more robust than each method by its own. This will possibly also hold for the context vector approach. Nevertheless, also in LSI and PLSI the highest improvements can be obtained on MED and the least on CRANFIELD.

Rungsawang & Rajman (1995) report results of the DSIR model for the CRANFIELD collection. Their model performed better than simple word-matching, but worse than the vector space model with IDF weights. Thus, our model outperforms DSIR, at least on CRANFIELD. Nevertheless, Rungsawang & Rajman do not employ term weights, which could certainly improve their results.

In Figures 2 and 3 we present some precision/recall curves for the four collections. We present the *probdiag_qcv_dcvmamd_idfdtfmvar* for the use of query context vectors and the best choice of *tf* or *bin* for the *intudiag* runs with *idfdcvmamd_idfdtfmamd* weights. The curve

for *vsm* is given for comparison. The precision values are interpolated recall level precision averages. This means that we calculated the interpolated precision at recall points 0, 0.1, 0.2, ..., 1 for each query and averaged the results over all queries in a collection. We used the same calculation method as employed in the TREC experiments (Vorhees & Harman, 1999).

Figure 2 shows that our method performs much better than the standard VSM with IDF weights on MED. The runs with the *intudiag* term correlation matrix work better than the runs with *probdiag*. This is because strong concept matching is very appropriate on MED. In fact, even slightly better results can be reported when the *probnodiag* correlation matrix is used with *bin* query encoding. The improvements over VSM on the CRANFIELD collection are smaller. There, the *probdiag* run is quite close to *vsm*. The precision values are better than those of *vsm* at recall levels higher than 0.1. For the *intudiag* run, the precision values at low recall levels are worse. This changes at recall level 0.2 from whereon the values are successively better than those of the other runs. The fact that stronger concept matching seems to perform better than *vsm* at higher recall levels can also be observed in the MED tests, where the improvement at these levels is even higher. This behavior seems obvious. Because of the characteristics of concept matching it will usually find more relevant documents earlier than *vsm*, but may have more noise in the first few documents found. In both collections we used *bin* instead of *tf* query vectors since they performed better. However, the results are quite similar for both methods.

The P/R curves for the CACM and CISI collections are shown in Figure 3. The results of our model are better than those of *vsm* except for *intudiag_tf_idfdcvmamd_idfdtfmamd* on CACM. As we mentioned before, this weighting combination is not a good choice for CACM. With respect to the CISI curves, the precision values are generally best for *intudiag_tf_idfdcvmamd_idfdtfmamd* and the effect of better precision values at higher recall

levels can be observed here as well.

Generally, the main advantage of the context vector model seems to be a performance improvement in retrieval tasks with vague or general information needs, or when high recall is required. It can also be expected that the model works better than VSM when the domain of the collection is rather wide. On the other hand, when the collection domain is small and queries are very precise, a direct word matching approach may be more suitable. Albeit some variants of the context vector model seem to work generally better than VSM (using the *probdiag* matrix definition), potentially higher improvement gains may be obtained by selecting an appropriate strategy for each collection.

The main disadvantage of the model is its memory and time consumption. The indexing process takes longer than in the classical VSM, because the term correlation matrix has to be generated. Furthermore, the method requires more memory since it has to store this matrix and the document context vectors. Even though, with a few modifications, the term context vectors could be stored in a symmetric matrix, the memory requirements increase with the number of index terms. This also holds for the document context vectors, where the number of non-zero elements in the document context vectors increases with the number of index terms. This is different in document vectors in VSM, which will usually have only a few non-zero elements, although they may be large in size.

However, the main problem regarding memory and time consumption is not in the indexing process, since this can be done in a batch mode. This issue is much more important in the retrieval process where short response times are usually required. If binary or term frequency query vectors are used, then query indexing is quite fast and does not require a lot of memory. In this case, the calculation of document/query distances by equation (5) will also be fast, because query vectors are sparse. The only remaining problem is the size of the

document context vectors. It will be difficult to load all vectors into memory in large collections. A solution to this problem is proposed by Rungsawang (1999) and Rungsawang & Rajman (1995). There, a subset of the set of index terms is selected before the actual indexing process. This subset spans a (reduced) vector space. The number of selected terms will usually be smaller than the number of index terms and thus, term, document and query context vectors are smaller in size.

We propose two other solutions. In the first one, the initial vector space is maintained, but only the most significant elements in each document context vector are stored and all others are set to 0. This does not actually reduce the dimensions, but generates sparser vectors with only a few non-zero elements. Experiments we did on MED, reducing the document context vectors to 100 non-zero element, show that the average precision over all relevant documents is generally still greater than 0.6. Another solution consists of calculating the document context vectors during the retrieval process. This is only feasible since the calculations can be “moved” into query indexing. Although this reduces the memory consumption for document representations, it increases the time for query indexing and requires the use of the term correlation matrix in this process.

7 Conclusions

In this paper, we presented our work on a context vector information retrieval model that uses term dependencies in the process of indexing documents and queries. The calculated context vectors that represent the documents of a collection are semantically richer descriptions of the issues treated in the documents. The use of such vectors means that the calculation of document/query similarity is based on a semantic-matching rather than on a simple word-matching approach. This brings more uncertainty or vagueness into the retrieval process, and, compared to standard VSM, the proposed model can be expected to improve

precision at high recall levels but may have slightly lower precision values at low recall levels. The results presented in this paper seem to confirm this hypothesis. We tested the model with different term dependency estimations and different term-weighting schemata on four test collections (MED, CRANFIELD, CACM, and CISI). The results are compared to the standard VSM with IDF weights and to a blind feedback approach using Rocchio's relevance feedback formula to automatically expand the original queries. The use of a concept space that has a direct correspondence to the original term space allows us to use query encodings based on binary, term frequency, or context vectors. Furthermore, it makes it possible, on one hand, to employ the term weighting techniques of standard VSM, and, on the other hand, it facilitates the definition of new term weighting schemata.

In general retrieval results gain from the incorporation of term dependencies. A relatively low concept matching approach works significantly better than VSM on all four collections. However, higher improvement gains can be obtained when different variants are used for each collection. Strong concept matching works best on MED. CRANFIELD requires an intermediate approach where concept matching is lowered down and direct word-matching increases. On CACM and CISI strategies that gives more importance to direct word matching and less to the dependencies among terms seem to perform best.

The empirical results tend to confirm that different variants of our method improve the retrieval performance for particular types of queries and/or document collections. We found that the model seems not very suitable for precise queries with just a few relevant documents. On the other hand, it works well for queries with many relevant documents. We think that such queries represent usually more general or vague information needs. In our future research we plan to analyze different types of queries and how it would be possible to identify those queries for which high improvements can be expected. We also think that context vector

comparison is more appropriate when the collection domain is rather wide, that is, the documents in the collection are semantically different.

Generally, we believe that different variants of our method should be used as complements for other models that work better on high precision tasks. We believe that a substantial improvement in retrieval performance goes along with a combination of different retrieval models. In this sense an IR system that combines various models should be able to select an appropriate retrieval strategy based on the characteristics of the underlying collection and the incoming query. Our future research will be centered in this issue.

Acknowledgements

Financial support for this study was provided in part by project grants FIS 97/0267 and CICYT TEL97-1073-C02-01 and by a FPI grant from the Madrid regional government.

References

Billhardt, H., Borrajo, D., & Maojo, V. (2000). Using Term Co-occurrence Data for Document Indexing and Retrieval. In Proceedings of the BCS-IRSG 22nd Annual Colloquium on Information Retrieval Research (IRSG'2000) (pp. 105-117). Cambridge: BCS-IRSG.

Bollmann-Sdorra, P., & Raghavan, V.V. (1998). On the Necessity of Term Dependence in a Query Space for Weighted Retrieval. Journal of the American Society for Information Science, 49, 1161-1168.

Buckley, C., Salton, G., Allan, J., & Singhal, A. (1995). Automatic Query Expansion using SMART: TREC-3. In D.K. Harman (Ed.), Overview of the Third Text REtrieval Conference (TREC-3) (pp.69-80). NIST Special Publication 500-226.

Chen, H., Tobun, D.N., Martinez, J., & Schatz, B. (1997). A Concept Space Approach to Addressing the Vocabulary Problem in Scientific Information Retrieval: An Experiment on the Worm Community System. Journal of the American Society for Information Science, 48, 17-31.

Chen, H., Yim, T., Fye, D., & Schatz, B. (1995). Automatic Thesaurus Generation for an Electronic Community System. Journal of the American Society for Information Science, 46, 175-193.

Deerwester, S., Dumais, S., Furnas, G., Landauer, T., & Harshman, R. (1990). Indexing by Latent Semantic Analysis. Journal of the American Society for Information Science, 41, 391-407.

Dennis, S.F. (1967). The Design and Testing of a Fully Automated Indexing-Searching System for Documents Consisting of Expository Text. In G. Schecter (Ed.), Information Retrieval: A Critical review (pp. 67-94). Thompson Book Company.

Efthimiadis, E.N. (1993). A User-Centered Evaluation of Ranking Algorithms for Interactive Query Expansion. In Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'93) (pp. 146-159). ACM.

Gotlieb, C.C, & Kumar, S. (1968). Semantic clustering of index terms. Journal of the ACM, 15(4), 493-513.

Hofmann, T. (1999). Probabilistic Latent Semantic Indexing. In Proceedings of the 22st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99) (pp. 50-57). ACM.

Jing, H., & Tzoukermann, E. (1999). Information Retrieval Based on Context Distance and Morphology. In Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99) (pp. 90-96). ACM.

Mitra, M., Singhal, A., & Buckley, C. (1998). Improving Automatic Query Expansion. In Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'98) (pp. 206-214). ACM.

Porter, M.F. (1980). An algorithm for suffix stripping. Program, 14, 130-137.

Raghavan, V.V., & Wong, S.K.M. (1986). A Critical Analysis of Vector Space Model for Information Retrieval. Journal of the American Society for Information Science, 37, 279-287.

Rocchio, J.J. (1971). Relevance feedback in information retrieval. In G. Salton (Ed.), The SMART Retrieval System—Experiments in Automatic Document Processing (pp. 313-323). Prentice Hall, Englewood Cliffs, NJ.

Rungsawang, A. (1999). DSIR: the First TREC-7 Attempt. In E.M. Vorhees, & D.K. Harman (Eds.), Proceedings of the Seventh Text REtrieval Conference (TREC-7) (pp. 425-

433). NIST Special Publication 500-242.

Rungsawang, A., & Rajman, M. (1995). Textual Information Retrieval Based on the Concept of the Distributional Semantics, In Proceedings of the third International Conference on Statistical Analysis of Textual Data (JADT'95), Rome, Italy.

Salton, G. (1989). Automatic Text Processing: The Transformation, Analysis and Retrieval of Information by Computer. Addison Wesley.

Salton, G., & Buckley, C. (1990). Improving retrieval performance by relevance feedback. Journal of the American Society for Information Science, 41, 288-297.

Salton, G., & McGill, M. (1983). Introduction to Modern Information Retrieval. McGraw Hill.

Salton, G., & Yang, C.S. (1973). On the specification of Term Values in Automatic Indexing. Journal for Documentation, 29, 351-372.

Schütze, H. (1992). Dimensions of Meaning. In IEEE Proceedings of Supercomputing 92.

Vorhees, E.M., & Harman, D.K. (Eds.). (1999). Proceedings of the Seventh Text REtrieval Conference (TREC-7) (Appendix A). NIST Special Publication.

Wong, S.K.M., Ziarko, W., Raghavan, V.V., & Wong, P.C.N. (1987). On Modeling of Information Retrieval Concepts in Vector Spaces. ACM Transactions on Database Systems, 12, 299-321.

		MED		CRANFIELD		CACM		CISI	
model specification		average precision	% over *vsm*	average precision	% over *vsm*	average precision	% over *vsm*	average precision	% over *vsm*
vsm		0.518		0.391		0.332		0.237	
vsm_bf		0.616	19 (5)	0.403	3 (1.61)	0.356	7 (1.14)	0.247	4.2 (1.63)
probdiag	*bin*	*idfdcvmamd_idfdcvmamd*		*dtfmvar_idftcvmamd*		*idfdcvmamd_idfdtfmvar*		*dtfmamd_idfdcvmvar*	
		0.579	11.8 (4.98)	0.405	3.7 (4.86)	0.332	-0.2	0.206	-13.2
	tf	*dcvmamd_idfdtfmvar*		*dtfmvar_idfdtfmamd*		*idfdcvmamd_idfdtfmvar*		*dcvmamd_idfdcvmvar*	
		0.563	8.7 (4.47)	0.401	2.5 (7.03)	0.36	8.2 (1.45)	0.258	9.1 (2.31)
	qcv	*dcvmamd_idfdcvmvar*		*dtfmamd_idfdtfmamd*		***dcvmvar_idfdtfmvar***		***dcvmamd_idfdcvmvar***	
		0.582	12.4 (5.34)	0.408	4.3 (4.04)	**0.381**	**14.6 (3.23)**	**0.263**	**11.2 (2.85)**
probnodiag	*bin*	***idfdcvmamd_idftcvmamd***		*idftcvmvar_idfdtfmamd*		*tcvmvar_idfdtfmamd*		*dtfmvar_idfdtfmamd*	
		0.665	**28.5 (5.94)**	0.393	0.5 (0.2)	0.311	-6.5	0.223	-5.9
	tf	*idftcvmamd_idfdcvmamd*		*idftcvmvar_idfdtfmamd*		*idftcvmvar_idfdtfmamd*		*idf_idfdtfmvar*	
		0.654	26.3 (5.87)	0.388	-0.9	0.334	0.4 (0.06)	0.25	5.4 (1.2)
	qcv	*dcvmamd_idfdtfmvar*		*idfdtfmamd_idfdtfmvar*		*dtfmamd_dcvmvar*		*dtfmvar_idfdcvmvar*	
		0.631	21.9 (3.69)	0.388	-0.7	0.289	-13	0.199	-15.8
intudiag	*bin*	*dcvmvar_dcvmvar*		*idftcvmvar_dtfmvar*		*idftcvmvar_idfdtfmamd*		*dtfmvar_idfdcvmamd*	
		0.665	28.4 (5.93)	0.418	7 (3.04)	0.321	-3.4	0.227	-4.1
	tf	*dcvmvar_idfdcvmamd*		*idfdtfmvar_no*		*idftcvmvar_dtfmvar*		*dcvmvar_idfdtfmvar*	
		0.650	25.7 (6.4)	0.415	6.1 (2.66)	0.342	3.1 (0.39)	0.26	9.9 (2.34)
	qcv	*dcvmamd_idftcvmvar*		***idfdcvmamd_idftcvmvar***		*dcvmamd_idftcvmvar*		*dtfmamd_idfdcvmvar*	
		0.644	24.4 (4.12)	**0.42**	**7.6 (3.09)**	0.354	6.6 (0.8)	0.233	-1.6

Table 1. Average precision results and relative improvement with respect to *vsm* for all four test collections. The best weight combinations are presented for the different term context vector calculations and query vector types used in the tests. The best overall result for each collection is presented in bold. In brackets we give the t-statistics.

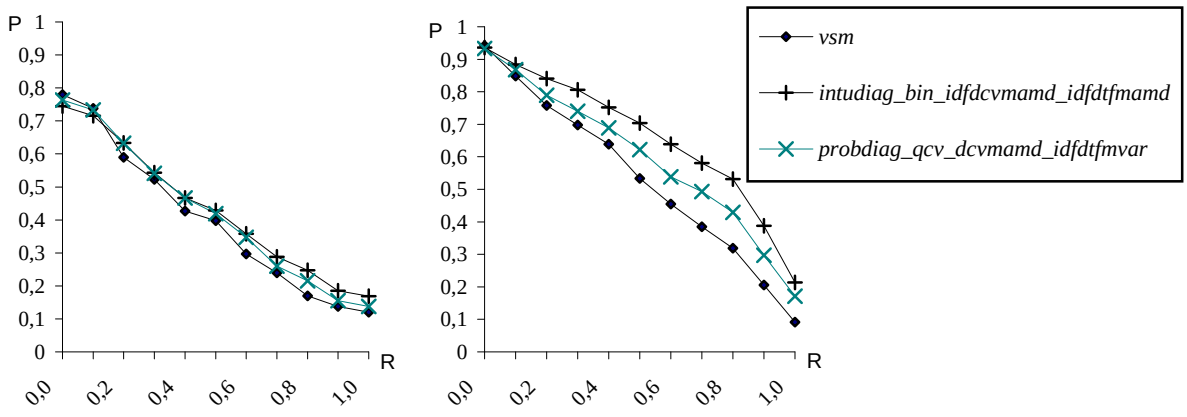

Figure 2. Precision/Recall Curves for CRANFIELD (left side) and MED (right curves)

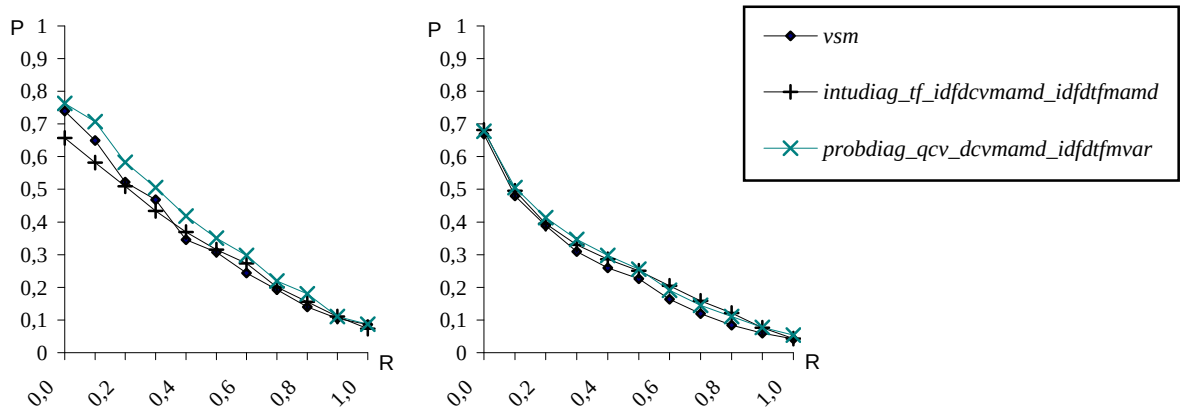

Figure 3. Precision/Recall Curves for CACM (left side) and CISI (right curves)