On Effective Conceptual Indexing and Similarity Search in Text Data

Charu C. Aggarwal
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598
charu@us.ibm.com

Philip S. Yu
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598
psyu@us.ibm.com

Abstract

Similarity search in text has proven to be an interesting problem from the qualitative perspective because of inherent redundancies and ambiguities in textual descriptions. The methods used in search engines in order to retrieve documents most similar to user-defined sets of keywords are not applicable to targets which are medium to large size documents, because of even greater noise effects stemming from the presence of a large number of words unrelated to the overall topic in the document. The inverted representation is the dominant method for indexing text, but it is not as suitable for document-to-document similarity search, as for short user-queries. One way of improving the quality of similarity search is Latent Semantic Indexing (LSI), which maps the documents from the original set of words to a concept space. Unfortunately, LSI maps the data into a domain in which it is not possible to provide effective indexing techniques. In this paper, we investigate new ways of providing conceptual search among documents by creating a representation in terms of conceptual word-chains. This technique also allows effective indexing techniques so that similarity queries can be performed on large collections of documents by accessing a small amount of data. We demonstrate that our scheme outperforms standard textual similarity search on the inverted representation both in terms of quality and search efficiency.

1. Introduction

In recent years, the large amounts of data available on the web has made effective similarity search and retrieval an important problem. Similarity indexing has uses in many web applications such as search engines or in providing close matches for user queries. A related problem is that of document-to-document similarity queries, in which the target is an entire document, as opposed to a small number of words for a specific user query. Such a system has considerable use in *recommender systems* in library or web applications, in which it is desirable to find the closest matches to a document which is currently being browsed. Other examples include personalized systems which can perform *information filtering*: a process which studies the long term access pattern by the user, and fetches the pages which are most consistent with his profile.

Similarity search is often used for short user queries in search engines such as $Yahoo!$, $Lycos$, $Google$ and $AltaVista$ [13] in which closest sets of matches are found to sets of keywords specified by the user. For such applications, the documents are represented in the form of an *inverted index* [11]. Other access methods [7] such as signature files exist, though the inverted representation seems to have become the method of choice in the information retrieval domain. The inverted representation consists of lists of Document Identifiers, one for each word in the lexicon. For each word w, its list contains all the document identifiers, such that the corresponding documents contain it. In addition, meta-information on word-frequency, position, or document length may be stored along with each identifier. For each user query, it suffices to examine the Document IDs in the inverted lists corresponding to the words in the query (or *target*). Details on how the similarity function is actually calculated for the relevant documents may be found in [11].

Similarity search has proven to be an interesting problem in the text domain because of the unusually large dimensionality of the problem as compared to the size of the documents. For example, a typical document collection on the world-wide web may have hundreds of thousands of words. This is significantly more than the average size of a document on the world-wide web. Considerable correlations between words exist because of synonymity and different descriptions of the same underlying latent concepts. Thus, two documents containing very different vocabulary could be similar in subject material. Similarly, two documents sharing considerable vocabulary could be topically very different. While applying the method to search engines (which is a special application of similarity search, in which the target document contains very few words), this problem is observed in the form of retrieval incompleteness and inaccuracy. For example, while querying on *cats* one may miss documents containing a description on the *feline* species, which do not explicitly contain the word *cat*. In many cases, the target may contain a multitude of concepts and subjects, which can be inferred only from the aggregate distribution of words in the document. Another well known problem is that of *polysemy*, in which the same word could refer to multiple concepts in the description (The word *virus* could refer to a computer virus, or to a biological virus.) Clearly, the ambiguity of the term can be resolved only by viewing it in the context of other terms in the document. In general, it is a challenging problem to design similarity functions for high dimensional applications [8] because of the fact that the aggregate behavior of high dimensional feature vectors contains a lot of information which cannot be inferred from individual attributes.

A well known method for improving the quality of similarity search in text is called *Latent Semantic Indexing* [6], in which the data is transformed into a new *concept space*. This concept space depends upon the document collection in question, since different collections would have different sets of concepts. Latent semantic indexing is a technique which tries to capture this hidden structure using techniques from linear algebra. The idea in LSI is to project the data into a small subspace of the original data such that the noise effects of synonymy and polysemy are removed. The advantageous effects of the conceptual representation extend to problems well beyond the text domain [2, 3]. A detailed description of the effects of conceptual representation may be found in [2, 6, 9].

LSI transforms the data from the sparse indexable representation (with the inverted index) in a very high overall dimensionality to a representation in the real space which is no longer sparse. Even though the new representation is of much lower overall dimensionality (typically about 200 or so dimensions are needed to represent the concept space), it is beyond the capacity of spatial indexing structures to handle effectively. Thus, we are presented with a difficult choice: if the data is represented in original format using the inverted index, it is less effective for performing document-to-document similarity search; on the other hand, when the data is transformed using latent semantic indexing, we have a data set which cannot be indexed effectively. Therefore, if we have a very large collection of documents, we would either be reduced to using a sequential scan in order to perform conceptual similarity search, or have to do with lower quality search results using the original representation and ignore the problems of synonymy and polysemy.

Another difficulty in performing document-to-document queries effectively is that a large fraction of the words in the target document are unrelated to the general subjects in it. Such words increase the noise effects, and reduce the likelihood of good search results. This is a problem that even latent semantic indexing cannot resolve very effectively. In addition to these issues, it is clear that the performance of the inverted representation of documents worsens with increasing number of words in the target document. In the search engine application it may be typical for a well-formulated query to contain words which are specialized enough, so that each of the relevant inverted lists contain only a small number of Document Identifiers. This is certainly not the case in a typical target document, where it is inevitable that some words may have substantially large inverted lists; yet cannot be ignored. Accessing these lists may significantly worsen the performance of an inverted representation when applied to a document-to-document query.

In this paper, we will discuss a technique which represents documents in terms of *conceptual word-chains*, a method which admits both high quality similarity search and indexing techniques. We will compare our technique to standard similarity search on the inverted index in terms of quality, storage, and search efficiency. We will show that the scheme achieves good qualitative performance at a low indexing cost. We note that our system is similar in spirit to the independently developed concept indexing technique [10]; which focusses on the dimensionality reduction problem; here we focus mainly on the efficiency and efffectiveness of the similarity indexing process.

This paper is organized as follows. The remainder of this section is devoted to related work, a formal description of the contributions of this paper, and some preliminary definitions and notations. In section 2, we show how to represent the documents in the concept space. In section 3, we discuss the indexing search technique used for the purpose of this paper. Section 4 discusses how the word-chains defining the concept space are created. The empirical results are illustrated in section 5, while section 6 contains the conclusions and summary.

1.1. Contributions of this paper

In this paper, we discuss a new method for conceptual similarity search for text using word-chaining which admits more efficient document-to-document similarity search than the standard inverted index, while preserving better quality of results. The technique also results in much lower storage requirements because it uses a compressed representation of each document. This low storage requirement in turn translates to higher search efficiency. Thus, we demonstrate that our scheme outperforms the standard similarity methods on text on all three measures: quality, storage, and search efficiency. This is also the first piece of work which treats the performance and quality issues of textual similarity search in one unified framework.

1.2. Preliminaries

In order to represent documents, we used the vector space representation in which each document is represented as a vector of words together with normalized term frequencies. Specifically, each document can be represented as a term vector of the form $\overline{a} = (a_1, a_2, \ldots a_n)$. Each of the terms a_i has a weight w_i associated with it, where w_i denotes the normalized frequency of the word. We used the standard $tf\text{-}idf$ normalization, in which less frequently occuring words in the aggregate collection are given higher weight. Details of the normalization process may be found in [11]. The *concatenation* of two documents is defined by appending one document with the other. Thus, if $\overline{a} = (a_1, \ldots a_n)$ be the weights in the vector space representation of document A, and $\overline{b} = (b_1, \ldots b_n)$ be the weights in the vector space representation of document B, then the concatenation of documents A and B is given by $(a_1 + b_1, \ldots a_n + b_n)$.

The similarity between two documents may be measured by using the *cosine* measure. The cosine similarity between two documents with weight vectors $U = (u_1, \ldots u_n)$, and $V = (v_1, \ldots v_n)$ is given by:

$$cosine(U, V) = \frac{\sum_{i=1}^{n} f(u_i) \cdot f(v_i)}{\sqrt{\sum_{i=1}^{n} f(u_i)^2} \cdot \sqrt{\sum_{i=1}^{n} f(v_i)^2}} \quad (1)$$

Here $f(\cdot)$ is a damping function such as the square-root or logarithm.

A *word-chain* is a set of closely related words along with a weight for each of the words. For example, a word-chain (along with corresponding weights) for military-related vocabulary could be:
army (75), troops (35), regiment (13), barracks (5),....
Typically, we expect the word-chain to contain a small number [1] of words which are semantically related to a given topic. This is somewhat similar to an automatically-generated thesaraus. For the purpose of our paper, we will treat a word-chain as a meta-document, and therefore all of the above notations and definitions for document representation and similarity are also applicable to a word-chain.

2. Defining the Conceptual Representation

There are a large number of objective functions such as the dice coefficient, jaccard coefficient, and cosine coefficient which are used to measure similarity between text documents [11]. There is not much consensus on the relative quality of the different ways of calculating similarity. The cosine function used in this paper is one of the more well known ones. However, what is common to all these techniques is that they are all susceptible to the redundancies, ambiguities and noise effects in textual descriptions.

In our paper, we change the representation of the document *at the attribute level* by defining new sets of attributes corresponding to concepts which have semantic significance. Each of these concepts is defined by a word-chain, which in turn is generated using the aggregate behavior of the document collection being indexed. For the time-being we will skim over the issue of how these word-chains are actually generated, and assume that each of them is a meta-document containing closely related words with weights representing relative frequency of occurance. More about word-chain generation will be discussed in a later section. Thus, a one-to-one correspondence exists between the new set of attributes defined and the word-chains. Let us assume that the number of such attributes is k^*. The larger the document collection available, the easier it is to generate sets of word-chains which reflect the different semantic concepts in the document collection. This is a useful property, since indexing is more critical for larger applications. Let us assume that the vector space representation of the set of of k^* word-chains are denoted by $\overline{v_1} \ldots \overline{v_{k^*}}$. Note that some of the word-chains may have overlapping vocabulary, corresponding to the fact that there may often be some relationship between the concepts in the collection. Let us also assume that $\overline{u}$ is the vector space representation of a given document. Then, in the conceptual coordinate system, the coordinate along the axis v_i is given by $max\{0, cosine(\overline{u}, \overline{v_i}) - t\}$. Here $t \in (0, 1)$ is a suitably defined constant which we shall define as the *activation threshold*. We will have more to say about this value later. The coordinate value thus calculated is the *conceptual strength*.

In order to intuitively understand the concept space, let us try to get a feel for what the coordinates corresponding to a given document refer to. Since each word-chain contains a set of semantically related words (or topical vocabulary), the conceptual strength of document with respect to a word-chain defines how closely the document matches this vocabulary. Thus, if a document contains a multitude of subjects which have sufficient presence, then this is reflected by non-zero conceptual strengths for each of the corresponding attributes. For example, a document X on a military hospital may share vocabulary from two of the word-chains out of the many attributes. These two attributes may correspond to the fact that it is a military related document, and that it is a document related to the medical profession. This description is more amenable to topical similarity search than a text description, since another document on military hospitals may not share a significant amount of text with X, and a document sharing a considerable amount of text with X unrelated to military hospitals may be a very poor

[1] As we will see in the empirical section, the typical size of a word-chain is about 50 words.

```
Algorithm Search(Target Concepts $C = \{c_1, \ldots c_r\}$,
        Strength: $\{t_1, \ldots t_r\}$)
begin
Initialize hash table as empty;
for each $i \in \{c_1, \ldots c_r\}$ do
  begin
  for each document $j$ indexed by concept $i$
      in the inverted representation do begin
    if entry for document $j$ does not exist in the hash table
       then create entry for $j$ in hash table and initialize entry value to 0;
    add $\frac{t_i \cdot q_{ij}}{\sqrt{L_j} \cdot \sqrt{L^t}}$ to the entry for document $j$ in the hash table;
    end;
  end;
Return the document with largest hash table entry;
end
```

Figure 1. Searching for the Closest Conceptual Neighbor

match. In the conceptual vector-space representation, the length of each vector is k^*, and the corresponding weights represent the conceptual strength. Let $A = (a_1, \ldots a_{k^*})$ and $B = (b_1, \ldots b_{k^*})$ be the coordinates of two points in the concept space. Then the conceptual cosine C-cosine between the two documents is defined in a similar way as the cosine function without the application of damping. Therefore, we have:

$$\text{C-cosine}(A, B) = \sum_{i=1}^{k^*} a_i \cdot b_i / \left(\sqrt{\sum_{i=1}^{k^*} a_i^2} \cdot \sqrt{\sum_{i=1}^{k^*} b_i^2} \right). \tag{2}$$

3. Indexing and Searching the Conceptual Representation

In the conceptual representation, when the word-chains are reasonably well-separated, a given document is likely to share substantial vocabulary with only a few of the word-chains. In terms of the coordinate representation, this means that only a few of the components are likely to be strongly positive, while most components are zero.

The definition of the conceptual strength introduced in the previous section ensures that when the similarity of a document to a word chain is less than a certain amount called the *activation threshold* (denoted by t), the corresponding conceptual strength for that coordinate is zero. The aim is to create a conceptual representation which summarizes the information in the document in terms of a small number of concepts which are specific to the particular collection in question. The use of an activation threshold ensures that even though a document may share a tiny amount of vocabulary from many word-chains, these correspond to the noise-effects stemming from words which are not generally related to the dominant topics in it. Furthermore, even though the document may share only a small amount of noisy vocabulary from each such word-chain, the sum of the noise effects over all the different word chains could be considerable. In effect, only a nicely filtered fraction of the vocabulary (corresponding to the dominant subjects) may be used in order to create the non-zero components in the conceptual representation. Such words are the *topical words* of that document.

One way of viewing conceptual representation is as a compressed representation of the documents which reduces the inherent noise effects of ambiguity, redundancy and unrelated vocabulary in a document. Latent semantic indexing achieves the noise reduction by picking a small subspace of the original data which shows the maximum variation in the distribution of the features. However, it does not achieve the compression effects since the documents are mapped to arbitrary numbers in the real domain. This also makes the data impossible to index because of the nature of the high dimensional representation.

Once the conceptual coordinates of each document have been determined, an inverted index of the new representation may be created. In this case, we have an inverted list for each of the k^* concepts corresponding to the word-chains. The document identifiers on a list are those documents which have a non-zero conceptual strength for the corresponding concept. The small number of positive components in the conceptual representation makes the data indexable, since the inverted representation is dependant upon the sparsity of the data. Along with each document ID j pointed to by each concept i, we also store the following information: (1) The *conceptual strength* q_{ij} for concept i in document j. (2) The *conceptual length* of document j, which is denoted by $L_j = \sum_{(\text{i: concept } i \in \text{ document } j)} q_{ij}^2$, and is one of the terms in the denominator of the conceptual cosine function.

Let $\{c_1, \ldots c_r\}$ be the set of concepts in the target document, and $\{t_1, \ldots t_r\}$ be the strength of the corresponding concepts. Our similarity search algorithm uses the meta-information in the inverted index [11]. Let L^t be the length of the target document. We say that a document is *touched* by the target, if it has at least one concept, which is in the target (In other words, its ID occurs on at least one of the inverted lists corresponding to that concept.). Then the algorithm builds a hash table of all documents which are touched by the target by examining the inverted lists one by one, and inserting document IDs in the hash table. When the document j from the inverted list for concept c_p ($1 \leq p \leq r$) is being examined, it keeps updating a similarity count entry for each document in the hash table by adding $\frac{t_p \cdot q_{c_p j}}{\sqrt{L_j} \cdot \sqrt{L^t}}$ to the corresponding hash table entry. At the end of the scan, for each document j which has been touched by the target, the value of the corresponding entry in the hash table will

```
**Algorithm** *ConceptualWordChains(Documents: D, NumberOfChains: $k^*$,
        Integer: ActivationThreshold)*
{ $\mathcal{W}$ = set of word chains; $\mathcal{C}_i$ = set of documents
    related to words chain $W_i \in \mathcal{W}$ }
{ $\mathcal{D}$ = set of all documents }
begin
  $\mathcal{W}$ = A randomly chosen set of $n_0$ documents;
  $ChainLength = StartChainLength$;
  while ($n_0 > k^*$)
  do begin
    $(\mathcal{D}, \mathcal{C}_1, \ldots \mathcal{C}_{n_0})$ =MatchDocuments($\mathcal{W}, \mathcal{D}, ActivationThreshold$);
    $\mathcal{W}$ =FindWordChains($\mathcal{C}_1, \ldots \mathcal{C}_{n_0}, ChainLength$);
    $\mathcal{W}$ =RemoveChains($\mathcal{W}, \mathcal{C}_1 \ldots \mathcal{C}_{n_0}$);
    $\mathcal{W}$ =ConsolidateChains($\mathcal{W}$, max$\{n_0 \cdot \gamma, k^*\}$);
    $n_0$ = max$\{n_0 \cdot \gamma, k^*\}$;
    $ChainLength$ = max$\{FinalChainLength, ChainLength * \theta\}$;
    { $\theta < 1$ indicates the rate at which the number of words
        in each chain reduces in an iteration }
  end;
end
```

Figure 2. Word-chain Creation

be given by $\sum_{(\text{p: concept } c_p \in \text{document } j)} \frac{t_p \cdot q_{c_p j}}{\sqrt{L_j} \cdot \sqrt{L^t}}$. This is the conceptual cosine (C-cosine) between the target and the document j. The search method is illustrated in the Figure 1. This method for calculating similarity directly from the meta-information in the inverted index can be adapted to textual similarity search on the original document for certain objective functions; though many IR systems use the inverted representation to identify the documents which should be accessed from the database, and then separately access the vector space representation in order to calculate similarity.

4. Generating the Conceptual Space

It is important to understand that the success of this method depends upon the finding of well separated word-chains which contain sets of closely correlated words. The concept of using word clustering for specific problem of text classification has been discussed by Baker and McCallum [5]. However, the applicability of that technique is restricted to the classification problems where the documents are already labelled. In our technique we concentrate on finding word-chains in an unsupervised way by iteratively creating word clusters and collections of documents associated with these clusters.

A good number of algorithms are known in the literature for performing text clustering [4, 1], though the focus of our technique is slightly different in terms of being able to find clusters on the *words* as opposed to the documents. Furthermore, there may be overlaps on the words in the different concepts. The method takes as input the activation threshold which is the user-defined limit on when a concept can be considered to be present in a document. This is the same value which is used during the process of building the inverted index and performing the similarity search. This activation threshold is useful in finding semantic relationships between the documents in the collection and the intermediate word-chains generated by the algorithm.

Our overall algorithm is an iterative one in which word-chains are used as representatives for grouping semantically related documents. In each iteration, the vocabulary in the word-chains is refined so that the resulting meta-document contains more and more tightly related sets of words. In the first iteration, each word-chain W_i is simply the set of words in an arbitrary document from the collection. The number of such chains is n_0. The exact value of n_0 is determined by the running time considerations, which will be discussed in a later section. At this point, closely related documents to each word-chain are found and collected in a set. Specifically, for word-chain W_i the set of semantically related documents to it is denoted by $\mathcal{C}_i$. A document may belong to multiple sets in $\{\mathcal{C}_1 \ldots \mathcal{C}_{n_0}\}$. These in turn are used in order to re-create each word-chain W_i by retaining the dominant vocabulary of $\mathcal{C}_i$. The number of words retained in each iteration is denoted by $ChainLength$. The algorithm starts with a larger value of $ChainLength$, and gradually reduces it in each iteration as the word-chains get more refined, and a smaller number of words are required in order to isolate the subject in a word-chain. This technique of finding closely related documents by iteratively refining both word-chains and document assignments to such chains is an effective technique for the creation of closely related groups of words. In each iteration, we keep consolidating chains which are close enough to belong to the same topic, so that we do not have concepts which are too similar in their vocabulary. At the same time we keep removing the chains which do not have enough matching documents, which corresponds to the fact that such concepts are not well represented in the aggregate behavior of the collection.

The overall algorithm for generation of word-chains is illustrated in Figure 2. We assume that the set of word-chains available to the algorithm at any stage is denoted by $\mathcal{W}$ and the documents which are being used in order to create the word-chains are denoted by $\mathcal{D}$. In each iteration, we assign a random sample $\mathcal{R}$ of the documents to their most semantically related word-chains in $\mathcal{W}$. The reason for picking a random sample $\mathcal{R}$ is that the assignment procedure requires $n_0 \cdot |\mathcal{R}|$ similarity functions calculations, where n_0 is the number of word-chains. This may dominate the running times. Therefore, picking a random sample size which is inversely proportional to the current number of word-chains n_0 balances the running time in each iteration. Furthermore, larger sample sizes are chosen in later iterations, when it is more critical to calculate word-chains in a more accurate and refined way. The particular value of the random sample used for our algorithm was $k^* \cdot |\mathcal{D}|/n_0$. This value of the random sample size chosen ensures that in the last iteration, when $n_0 = k^*$, the entire document collection is used. A document is assigned to all the word-chains in $\mathcal{W}$, to which the similarity of the document is larger than the activation threshold. Thus, a document could get assigned to multiple chains, corresponding to the different subjects in it. At the end of this procedure, the sets of documents $(\mathcal{C}_1, \ldots \mathcal{C}_{n_0})$ are returned, which corresponds to the different word-chains.

After the assignment procedure, we create a word-chain W_i out of each group $\mathcal{C}_i$. To do so, we concatenate the documents in each semantically related group which was created in the assignment process, and we project out the words with the least weight from the corresponding meta-document. This ensures that only the terms which are frequently occuring within that group of semantically related documents are used in the word-chain. The number of terms in the word-chains reduces by a geometric factor θ in each iteration. We shall refer to this value θ as the *projection factor*. This is because in the first few iterations, when the word-chains are not too refined, a larger number of dimensions need to be retained in the projection in order to avoid premature loss of information. In later iterations, the word-chains become more refined and it is possible to project down to a fewer number of words.

In each iteration, we reduce the number of word-chains by the *chain-consolidation factor* γ by consolidating very closely related chains. This reduces the likelihood of redundancy in the semantic representation of the documents, when expressed in terms of the final word chains (similar to the problem of synonymy). The merging process is implemented using a simple single linkage clustering method. Each word-chain is represented by a node in an undirected graph, and the similarity between each pair of word-chains is calculated. Edges are added to the graph in decreasing order of similarity, until there are $n_0 \cdot \gamma$ connected components. The new reduced set of word-chains are then re-created by concatenating the chains corresponding to each such component. Although the simple linkage process is somewhat naive, it is very fast and effective for values of the consolidation factor $\gamma \geq 0.1$. This is because single-linkage methods are very effective for cases when a small number of merges are performed before recalculation of nearest neighbors. The projection factor θ was picked in a way, so that the reduction of the initial chain length to the final chain length occured in the same number of iterations as the reduction of the number of word-chains from n_0 to k^*. In order for this to happen, the following relationship must be satisfied: $\log_\gamma(n_0/k^*) = \log_\theta(FinalChainLength/StartChainLength)$.

Some of the word-chains may turn out to be meaningless and consist of unrelated words. Such word-chains attract very few documents from the original set during the matching process. In order to find which word-chains should be removed, we calculated the mean μ and standard deviation σ of the distribution of documents in the groups $\mathcal{C}_i$ for $i \in \{1, \ldots n_0\}$. We discard all those word-chains W_i from $\mathcal{W}$ such that the number of documents in $\mathcal{C}_i$ is less than the threshold value of $\mu - r \cdot \sigma$. Here r is a parameter, which defines the statistical level of significance at which one would like to remove the poorly-defined chains.

5. Empirical Results

We used a scan of the $Yahoo!$ taxonomy from November 1996. This taxonomy contained a total of 167193 documents, over a lexicon of approximately 700,000 words. Each document in this taxonomy contained an average of about 80 to 100 distinct words from the lexicon. The very commonly occuring terms (or stop words) had already been removed from this lexicon. In addition, we removed words which rarely occured in the collection. Specifically, any words which occured in the entire collection less than 6 times was removed. Most of these words turned out to be misspellings or creative variations of ordinary words, which had no significance in the similarity indexing process. The resulting lexicon contained approximately 80,000 words.

Since the inverted representation is the primary data structure which provides indexing capabilities for text, we compared our scheme to it. Most of the results in this paper compare the widely used cosine textual function and its conceptual analogue (see Equations 1 and 2). For the textual cosine, the square-root damping function[2] was used and the inverse-document-frequency (idf) normalization [11] was used in order to weight the importance of the different words. We evaluated the textual similarity function against the conceptual method for both quality and efficiency.

5.1. Quality

This is difficult to measure because of its inherently non-quantitative nature. Therefore, we will provide evidence of the effectiveness of the conceptual technique in an indirect way. We will use the class labels on the $Yahoo!$ data set in order to check how well the retrieved results matched the target. If it is assumed that the manually defined $Yahoo!$ class labels reflect topical behavior well, then the relationship between the class labels of the target and search results can provide very good evidence of search quality. We found that the conceptual technique was able to match the documents well in terms of overall subject matching, whereas the textual representation was often mislead by the noise effects of individual words. Since

[2]The square-root damping function was slightly better than the logarithmic damping function in our empirical tests.

Target Document	**Textual Neighbor**	**Conceptual Neighbor**
@Arts@Architecture@Architects	@Arts@Architecture@Architects (16%) @Arts@Architecture (24%) @Arts (36%)	@Arts@Architecture@Architects (34%) @Arts@Architecture (47%) @Arts (58%)
@Arts@Art_History@Artists	@Arts@Art_History@Artists (16%) @Arts@Art_History (29%) @Arts (38%)	@Arts@Art_History@Artists (29%) @Arts@Art_History (44%) @@Arts (49%)
@Arts@Dance	@Arts@Dance (19%) @Arts (28%)	@Arts@Dance (44%) @Arts (55%)
@Government@Military@Air_Force	@Government@Military@Air_Force (18%) @Government@Military (29%) @Government (36%)	@Government@Military@Air_Force (33%) @Government@Military (48%) @Government (51%)
@Health@Nursing	@Health@Nursing (19%) @Health (29%)	@Health@Nursing (31%) @Health (48%)
@Recreation@Sports@Tennis	@Recreation@Sports@Tennis (17%) @Recreation@Sports (29%) @Recreation (30%)	@Recreation@@Sports@Tennis (32%) @Recreation@Sports (45%) @Recreation (48%)
@Science@Biology@Botany	@Science@Biology@Botany (18%) @Science@Biology (25%) @Science (31%)	@Science@Biology@Botany (32%) @Science@Biology (42%) @Science (51%)
Overall Average **20 categories**	Lowest level class (16%) One level higher (28%)	Lowest level class (28%) One level higher (47%)

Table 1. Matching of the subjects in target and nearest neighbors

the focus of this paper is in measuring the quality and coherence of the nearest neighbor obtained from conceptual indexing, we need some hard criterion for quantification of such qualitative measurements. To do so, we used the $Yahoo$! class labels associated with the documents. We compared the class labels in the retrieved results to the class labels of all levels of the $Yahoo$! hierarchy in the target. Specifically, we computed the $p = 20$ closest neighbors to the target, and calculated the percentage of neighbors from the same or a hierarchically related class. The results are presented in Table 1. The first column indicates the class label of the target document; whereas the second and third columns indicate some statistics of the class distributions of the search results for the textual and conceptual neighbors respectively for all levels of the $Yahoo$! hierarchy which are related to the target. For example, for a target document in the @Arts@Architecture@Architects category, we would try to find the percentage of neighbors belonging to each of the subtrees of $Yahoo$! corresponding to @Arts@Architecture@Architects, @Arts@Architecture, and @Arts respectively. Clearly the percentage of matching neighbors would always be higher while trying to make a partial match with a hierarchically related node. The values reported in each entry of Table 1 are determined by averaging over all targets in the corresponding $Yahoo$! class. It is also apparent that it is goodness for these accuracy numbers to be as high as possible, if we assume that $Yahoo$! class labels reflect topical behavior well.

As illustrated in Table 1, an exact match between the class labels of the target document and the nearest neighbors was found a very small percentage of the time for the textual nearest neighbor. We note that we are only using the matching percentage of class labels of an unsupervised similarity search procedure in order to demonstrate the qualitative advantages of conceptual similarity. (Therefore the results from supervised classifiers are a bound on the kind of accuracy one could hope to qualitatively achieve with a simple unsupervised nearest neighbor technique.) The accuracy of the textual neighbor ranged in the vicinity of 14-19% over the different categories. On the other hand, the accuracy for the conceptual nearest neighbor ranged between 30-40%. The accuracy numbers varied considerably over the different categories. In each case, the conceptual neighbor always dominates the effectiveness of the textual neighbor. The test results for the hierarchical relationship of the retrieved documents to the target were similar. In these cases the percentage accuracies for the textual nearest neighbor varied between 25-29%, whereas the numbers for the conceptual nearest neighbor were in the range of 45-50%. Furthermore, the conceptual neighbor performed much better in each of the individual cases.

5.2. Storage Requirements and Search Efficiency

On generating the conceptual space, we found that each document contained about 5-10 concepts. On the other hand, the average document contained about 100 words. This is a dramatic reduction in the space required to store and index documents. This results in savings both in terms of storage efficiency and computational efficiency of disk

Method	Result
Textual	426 676 (IDs Accessed)= 5.1 MB
Conceptual	9032 (IDs Accessed)= 108 KB

Table 2. Search Efficiency for the two methods

accesses. The original set of 167,000 documents required a total of about 87.7 MB in the inverted representation, whereas the conceptual representation required only 8.3 MB. This translates to more than an order of magnitude improvement in the space required for storage. These improved storage requirements also translate to greater efficiency during query time in two ways: **(1)** In the textual representation, some of the inverted lists are very long because of relatively commonly occuring words (which are not stop words). These inverted lists contribute to a substantial portion of the access time. This is never the case in the conceptual representation. **(2)** Each of the target documents contains a fewer number of concepts; so fewer number of the inverted lists need to be accessed.

Since the inverted representation is used by both the schemes, it follows that the search efficiency may be determined easily by calculating the number of document IDs (including repetitions) which were accessed by each of the methods. The number of ID accesses and corresponding disk access requirements are illustrated in Table 2. The numbers are averaged over 1000 queries for each case. As we see from Table 2, each similarity search query in the textual representation required an access of 426,676 documents IDs from the inverted index, which sums to a total of about 5.1 MB. This is an exceptionally high access requirement for a *single query*. Note that the number of document IDs accessed is more than the number of documents in the collection, corresponding to the fact that the same ID occured on multiple lists. The number of distinct IDs accessed for each query averaged at about 10% of the total documents. The reason for this exceptionally large number of documents being accessed is that some words are more frequent in the document collection than others (but are important enough to be not considered stop-words). The lists for these words add significantly to the access cost. The use of the inverted representation ensures that the search efficiency requirements scale linearly with the size of the document collection. Thus, an access requirement of 5.1 MB per query for a moderately sized collection of 167,000 documents does not bode well for the performance on much larger collections. In contrast, the conceptual indexing technique requires the access of an average of 9032 document IDs per query. This is about 108 KB per query and more than an order of magnitude improvement over the textual indexing technique.

6. Conclusions and Summary

In this paper, we discussed a new method for conceptual indexing and similarity search of text. The techniques discussed in this paper can be used for dramatically improving the search quality as well as search efficiency. Although our technique is designed with a focus on document-to-document similarity queries, the techniques are also applicable to the short queries of search engines. This work provides an integrated view of qualitatively effective similarity search and performance efficient indexing in text; an issue which has not been addressed before in this domain.

References

[1] C. C. Aggarwal, S. C. Gates, P. S. Yu. On the Merits of Using Supervised Clustering for building Categorization Systems. *ACM SIGKDD Conference*, 1999.

[2] C. C. Aggarwal. On the Effects of Dimensionality Reduction on High Dimensional Similarity Search. *ACM PODS Conference*, 2001.

[3] C. C. Aggarwal, S. Parthasarathy. Mining Massively Incomplete Data Sets by Conceptual Reconstruction. *ACM KDD Conference*, 2001.

[4] P. Anick, and S. Vaithyanathan. Exploiting Clustering and Phrases for Context-based Information Retrieval. *ACM SIGIR Conference*, 1997.

[5] L. Douglas Baker, A. K. McCallum. Distributional Clustering of words for Text Classification. *ACM SIGIR Conference*, 1998.

[6] Dumais S., Furnas G., Landauer T. Deerwester S., Using Latent Semantic Indexing to improve information retrieval. *ACM SIGCHI Conference, 1988*.

[7] C. Faloutsos Access Methods for Text. *ACM Computing Surveys* 17, 1, March 1995.

[8] A. Hinneburg, C. C. Aggarwal, D. A. Keim. What is the Nearest Neighbor in High Dimensional Spaces? *Proceedings of the VLDB Conference*, 2001.

[9] J. Kleinberg, A. Tomkins. Applications of Linear Algebra in Information Retrieval and Hypertext Analysis. *ACM PODS Conference*, 1999.

[10] G. Karypis, E.-I. Han. Concept Indexing: A Fast Dimensionality Reduction Technique with Applications to Document Retrieval and Categorization. *CIKM Conference*, 2000.

[11] G. Salton, M. J. McGill. Introduction to Modern Information Retrieval. *Mc Graw Hill*, New York, 1983.

[12] H. Schutze, C. Silverstein. Projections for efficient document clustering. *ACM SIGIR Conference*, 1997.

[13] http://www.altavista.com, http://www.lycos.com, http://www.yahoo.com http://www.google.com