Under consideration for publication in Knowledge and Information
Systems

Concept Hierarchy Based Text Database Categorization

Weiyi Meng[1], Wenxian Wang[1], Hongyu Sun[1], Clement Yu[2]

[1]Department of Computer Science
SUNY at Binghamton, Binghamton, NY 13902, meng@cs.binghamton.edu
[2]Department of Computer Science
University of Illinois at Chicago, Chicago, IL 60607, yu@eecs.uic.edu

Abstract. Document categorization as a technique to improve the retrieval of useful documents has been extensively investigated. One important issue in a large-scale metasearch engine is to select text databases that are likely to contain useful documents for a given query. We believe that database categorization can be a potentially effective technique for good database selection, especially in the Internet environment where short queries are usually submitted. In this paper, we propose and evaluate several database categorization algorithms. This study indicates that while some document categorization algorithms could be adopted for database categorization, algorithms that take into consideration the special characteristics of databases may be more effective. Preliminary experimental results are provided to compare the proposed database categorization algorithms. A prototype database categorization system based on one of the proposed algorithms has been developed.

Keywords: Database Categorization; Concept hierarchy; Metasearch; Database selection

1. Introduction

The Internet has become a vast information source in recent years. To help ordinary users find desired data in the Internet, many search engines have been created. Each search engine has a corresponding database that defines the set of documents that can be searched by the search engine. Usually, an index for all documents in the database is created and stored in the search engine to speed up the processing of user queries. For each term, which represents a content word or

Received xxx
Revised xxx
Accepted xxx

a combination of several (usually adjacent) content words, this index can identify the documents that contain the term quickly.

Although general-purpose search engines that attempt to provide search capabilities for all documents on the Web, like Excite, Lycos, HotBot, and AltaVista, are quite popular, most search engines on the Web are special-purpose search engines that focus on documents in confined domains such as documents in an organization or of a specific subject area. Tens of thousands of special-purpose search engines exist in the Internet. The information needed by a user is frequently stored in the databases of multiple search engines. As an example, consider the case when a user wants to find research papers in some subject area. It is likely that the desired papers are scattered in a number of publishers' and/or universities' databases. It is very inconvenient and inefficient for the user to determine useful databases, search them individually and identify useful documents all by him/herself. A solution to this problem is to implement a *metasearch engine* on top of many local search engines. A metasearch engine is system that can provide unified access to multiple existing search engines. It does not maintain its own index on documents. But a sophisticated metasearch engine may store characteristic information about the contents of underlying search engines in order to provide better service. When a metasearch engine receives a user query, it first passes the query (with necessary reformatting) to the appropriate local search engines, and then collects (sometimes, reorganizes) the results from its local search engines. Clearly, with such a metasearch engine, the above user's task will be drastically simplified.

A substantial body of research work addressing different aspects of building an effective and efficient metasearch engine has been accumulated in recent years. One of the main challenging problems is the *database selection problem*, which is to identify, for a given user query, the local search engines that are likely to contain useful documents (Baumgarten, 1997; Callan et al, 1995; Dreilinger and Howe, 1997; Gravano and Garcia-Molina, 1995; Kahle and Medlar, 1991; Koster, 1994; Liu et al, 2001; Manber and Bigot, 1997; Meng et al, 1998; **?**; Selberg and Etzioni, 1997; Yu et al, 1999; Yuwono and Lee, 1997). The objective of performing database selection is to improve efficiency as it enables the metasearch engine to send each query to only potentially useful search engines, reducing network traffic as well as the cost of searching useless databases.

Most existing database selection methods rank databases for each query based on some quality measure. These measures are often based on the similarities between the query and the documents in each database. Similarities are computed based on the match of terms in the query and documents. For example, a measure used in gGlOSS (Gravano and Garcia-Molina, 1995) to determine the quality of a database with respect to a given query is the sum of the similarities between the query and highly similar documents in the database when the similarity is greater than or equal to a threshold. As another example, we have developed a scheme to rank databases optimally for finding the m most similar documents across multiple databases with respect to a given query for some integer m. The ranking of the databases is based on the estimated similarity of the most similar document in each database. Our experimental results indicate that on the average more than 90% of the most similar documents will be retrieved by our method (Yu et al, 1999). Studies in information retrieval indicate that when queries have a large number of terms, there is a strong correlation between highly similar documents and relevant documents provided appropriate similarity functions and term weighting schemes, such as the Cosine function and tf*idf weight formula (Salton and McGill, 1983), are used. However, for queries that are short, the above correlation is weak. The reason is that for a long query, the terms in the query provide context to each other to help disambiguate the meanings of different terms. In a short query, the particular meaning of a term often cannot be identified correctly. Queries submitted by users in the Internet environment are usually very short and the average number of terms in a typical Internet query is only 2.2 (Jansen et al, 1998; Kirsch, 1998). In summary, a similar document to a short query may not be useful to the user who submitted the query because the matching terms may have different meanings. In general the retrieval effectiveness of search engines need to be improved.

Several techniques have been developed to remedy the above problem. The first technique is a *query expansion* based method. The idea is to add appropriate terms to a query before it is processed. In (Xu and Callan, 1998), a training database that has a similar coverage of subject matters and terms as the set of actual databases is utilized. Upon receiving a query, the training database is searched; terms are extracted and then added to the query before retrieval of documents from the actual databases takes place. In the Internet environment where data are highly heterogeneous, it is unclear whether such a training collection can in fact be constructed. Even if such a collection can be constructed, the storage penalty could be very high in order to accommodate the heterogeneity. The second technique is to use linkage information among documents to determine their ranks (degrees of importance) and then incorporate the ranks into the retrieval process. Links among documents have been used to determine the popularity and authority of Web pages (Kleinberg, 1998; Page et al, 1998). In (Yu et al, 2000), a weighted sum of document similarity and link-determined document rank is used to determine the *degree of relevance* of a document. The idea is that among documents whose similarities with a given query are about the same, the one with the highest rank is likely to be most useful. Based on this intuition, the method in (Yu et al, 2000) is extended to rank databases according to the degree of relevance of the most relevant document in each database. The third technique is to associate databases with concepts. When a query is received, it is mapped to a number of concepts and then those databases associated with those mapped concepts are searched. The concepts associated with a database are used to provide some contexts for terms in the database. As a result, the meanings of terms can be more accurately determined. In (Manber and Bigot, 1997), each database is manually assigned to one or two concepts. When a user submits a query, the user also specifies the related concepts. In (Fan and Gauch, 1999), training queries are used to assign databases to 13 categories/concepts. These methods may not scale to a large number of databases due to their manual or training nature. In (Xu and Croft, 1999), several methods are proposed to assign clusters/databases to topics/concepts. Two methods (global clustering and local clustering) require that documents be physically regrouped (based on K-Means clustering) into clusters. The third method (multiple-topic representation) in (Xu and Croft, 1999), while keeping each database as is, requires each local system to logically cluster its documents so that a database can be assigned to the topics associated with its clusters. These approaches may not be practical in the Internet environment as substantial cooperation is required across autonomous local systems. In contrast, the methods we propose in this paper are based on existing databases and they do not require document clustering.

In this paper, we study methods to assign databases to concepts. This paper has the following contributions.

1. A concept hierarchy is utilized for database categorization. The methods in (Fan and Gauch, 1999; Manber and Bigot, 1997) assign databases to a flat space of concepts. In contrast, we assign databases to concepts that are hierarchically organized. While there have been reports on categorizing documents according to a concept hierarchy to improve retrieval performance (for example, see (Wang et al, 1999; Yang and Chute, 1994)), we are not aware of any existing work for categorizing databases utilizing a concept hierarchy.
2. Two new methods are proposed to assign databases to concepts. While one of them is extended from a method for document categorization, the other is only applicable for database categorization. Both methods assign databases to concepts fully automatically.
3. Experiments are carried out to compare these two new methods with a much more complex method (a Singular Value Decomposition (SVD) based method (Yang and Chute, 1994)). Our experimental results indicate that one of our fully automatic methods performs very well and outperforms the other two methods. Based on one of our proposed database categorization algorithms, a prototype system has been developed to automatically categorize search engines.

The rest of the paper is organized as follows. In Section 2, we describe the concept hierarchy used in this research. In Section 3, we describe three methods for categorizing databases based on a concept hierarchy. In Section 4, we report our experimental results. In Section 5, we describe a prototype search engine categorization system. We conclude the paper in Section 6.

2. Concept Hierarchy

The concept hierarchy contains a large number of concepts organized into multiple levels such that concepts at higher levels have broad meanings than those at lower levels. In general, a child concept is more specific in meaning than its parent concept. With such a concept hierarchy, we can assign different search engines to appropriate concepts in the hierarchy. Ideally, if the database of a search engine contains good documents relating to a given concept, then the search engine should be assigned to the concept. It is possible for the same database to be assigned to multiple concepts. Different methods for assigning databases to concepts in the concept hierarchy will be discussed in Section 3.

The concept hierarchy and its associated search engines could be used in the metasearch engine environment as follows. When a user needs to find some documents from the Web, the user first browses the concept hierarchy starting from the root. This process is very much like browsing the Yahoo category hierarchy. After the user identifies a set of concepts that best matches his/her needs, the user then submits a query against the metasearch engine. The metasearch engine can now select the databases to search in two steps. In the first step, a preliminary selection is performed based on the user-identified concepts. Specifically, if only one concept is identified by the user, then the databases associated with the concept are selected; if multiple concepts are identified, then two options are possible: (1) the databases that are common to all the identified concepts are selected, and (2) the databases that are associated with at least one identified concept are selected. While the first option may likely produce better databases, it may not produce enough databases. The second option is likely to produce more databases. In the second step, a regular, say similarity-based, database selection method (such as those proposed in (Gravano and Garcia-Molina, 1995; Yu et al, 1999; Yu et al, 2001)) is used to further select the best databases for the query from the returned databases by the first step. The databases selected by this two-step approach are much more likely to contain relevant documents to the query than those selected using the second step only. The exact performance gain of this two-step approach over the normal one-step approach, however, will be studied in a future research and will not be reported in this paper.

We now discuss how the concept hierarchy can be constructed. We would like the hierarchy to have the following features. First, it must reflect the breadth of the topics available in the Internet, as the concept hierarchy will be used to support the search of documents on the Web. Second, concepts should be properly placed in the hierarchy, that is, parent-child relationships among different concepts should be appropriate. Instead of constructing a concept hierarchy from the scratch, we decide to utilize a well-known category hierarchy for organizing documents, i.e., the Yahoo category hierarchy. However, the category hierarchy in Yahoo has too many levels and the concepts at lower levels are too specific for categorizing databases. In this project, we decide to "borrow" only the first two levels of the Yahoo category hierarchy. A simple program is written to automatically fetch the first two levels from the Yahoo category hierarchy. Some manual adjustment of the concept hierarchy is made to improve the quality of the hierarchy for our application. For example, some of Yahoo's second level categories include topics like "By Region", "Browse by Region", etc. These are not considered to be very useful for us and therefore pruned. An advantage for "borrowing" the Yahoo category hierarchy is that many Internet users are already familiar with it.

In order to assign databases to concepts automatically, we need to generate a text description for each concept. We could use the term(s) or phrase(s) representing a concept as the description of the concept. But such a description may not be sufficient to convey the meaning of the concept as each concept uses only one or two terms/phrases. A longer description is desired. Manually providing the description for each concept can be time-consuming and the quality of the description cannot be guaranteed. Again, we decide to utilize the information in the Yahoo category hierarchy to automatically generate the description for each concept in our concept hierarchy. Our approach can be sketched as follows. Each concept has a number of child concepts and these child concepts together cover different aspects of the parent concept. Based on this observation, we use the set of terms that appear in all child concepts of a given concept as the description of the concept. Stop word removal and stemming are applied to the description. Note that in order to generate the description for a second level concept in our hierarchy, the child concepts of the concept in the Yahoo category hierarchy are needed. This means that even though our concept hierarchy has only two levels, the hierarchy and the corresponding descriptions are generated from the top three levels of the Yahoo category hierarchy.

3. Database Categorization Algorithms

In this section, we present three methods for assigning databases to concepts in the concept hierarchy. In all the three methods, if a database is assigned to a concept, then the database is also assigned to its parent concept. The rationale is that if a database is useful for a child concept then it is also useful to the parent concept. However, it is possible for a database to be assigned to a parent concept but to none of its child concepts. This is because a parent concept is usually much broader than any of its child concepts.

3.1. High Similarity with Database Centroid (HSDC)

The database D of each local search engine has a set of documents. Each document d in D can be represented as a vector of terms with weights, i.e., $d = (d_1, \ldots, d_n)$, where d_i is the weight of term $t_i, 1 \leq i \leq n$, and n is the number of distinct terms in D. Suppose each d_i is computed by the widely used $tf * idf$ formula (Salton and McGill, 1983), where tf represents the term *frequency weight* of the term (t_i) and idf represents the *inverse document frequency* weight of the term. From the vectors of all documents in D, the **centroid**, denoted $c(D)$, of database D can be computed. The centroid $c(D)$ is also a vector of n dimensions, $c(D) = (w_1, \ldots, w_n)$, where w_i is obtained by first adding the weights of t_i in all documents in D and then dividing the sum by the number of documents in D, $1 \leq i \leq n$.

In Section 2, we discussed how to represent each concept using a description. Note that a description is essentially a document. Therefore, the set of descriptions for the concepts at the same level can be treated as a document collection. Now each description can be represented as a vector of terms with weights, where each weight is computed using the $tf * idf$ formula based on this description collection.

The similarity between a concept description and a centroid can be computed using the Cosine similarity function, which is basically the inner product of the two corresponding vectors divided by the product of the norms (lengths) of the two vectors. Our first strategy for assigning a database D to concepts in the concept hierarchy is based on the similarities between the database and the concepts. This strategy, HSDC, can be described as follows: First, compute the similarity between c(D) and all concepts in all levels of the concept hierarchy and sort the concepts in descending similarity values. Second, if a database is to be assigned to k concepts, then the k concepts that have the largest similarities with the database will be used.

This strategy follows the retrieval model in standard information retrieval: if a user wants to retrieve k documents for his/her query, then the k documents that are most similar to the query are retrieved. There is, however, a slight complication in our database categorization problem. Recall that when a database D is assigned to a concept and the concept has ancestors, then D will also be assigned to the ancestors of D. This means that a database may actually be assigned to more then k concepts. With our two-level concept hierarchy, a concept may have at most one ancestor concept (but many child concepts may share the same ancestor), implying that D may actually be assigned to as many as $2k$ concepts. If we insist that D cannot be assigned to more than k concepts, then the following modified strategy can be used. Find the concept, say C, which has the next largest similarity with D (in the first iteration, the concept with the largest similarity will be chosen). Let k_1 be the number of concepts to which D has already been assigned and let k_2 be the number of concepts in the set containing C and all ancestors of C to which D has not been assigned. If $k_1 + k_2 < k$, then assign D to C as well as to those ancestors of C to which D has not been assigned; otherwise, namely if $k_1 + k_2 > k$, then stop. Repeat this process until the stop condition is satisfied.

One related question is how to determine the number k in practice. One way could be as follows. First, manually assign a small number of databases to concepts. Next, compute the average number of concepts each database is assigned to. This average number can then be used as the k to assign new databases.

A variation of the above strategy for assigning databases is to use a similarity threshold T instead of a number k. In this variation, a database is assigned to a concept if the similarity between the database and the concept is greater than or equal to T. This variation has the advantage of avoiding assigning a database to a concept with a low similarity. On the other hand, determining an appropriate T may not be easy in practice.

Applying this method to categorize search engines in practice may not be easy. The difficulty is due to the fact that this method requires the computation of database centroids. The accurate centroid of a database can only be obtained if all documents of the database are accessible and it may be difficult to identify and access all documents in a search engine. A possible solution is to sample documents of each search engine and use the sampled documents to compute an approximate database centroid.

3.2. Singular Value Decomposition (SVD)

Singular Value Decomposition (SVD) (Furnas et al, 1988) has been used as an effective method for document categorization (Yang and Chute, 1994). In this section, we apply this method for database categorization.

We first review the method in (Yang and Chute, 1994) for assigning a set of documents to a set of categories/concepts. Note that the same document may be assigned to a number of concepts. Let matrix A represent the set of input documents and matrix B represent an assignment of the input documents to the concepts, respectively.

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

$$B = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1k} \\ b_{21} & b_{22} & \cdots & b_{2k} \\ \cdots & \cdots & \cdots & \cdots \\ b_{m1} & b_{m2} & \cdots & b_{mk} \end{pmatrix}$$

A row in matrix A is a document and a_{ij} denotes the weight of term t_j in the ith document, $i = 1, \ldots, m, j = 1, \ldots, n$. The ith row in B represents the assignment of the ith document to the concept set and b_{ij} denotes the extent to which the ith document is assigned to the jth concept, $i = 1, \ldots, m, j = 1, \ldots, k$. As an example, if b_{ij} takes only binary values, then $b_{ij} = 1$ indicates that the ith document is assigned to the jth concept and $b_{ij} = 0$ indicates that the ith document is not assigned to the jth concept. In general, b_{ij} may take non-binary values. Matrix B is obtained from a known (manual) assignment of documents to the concept set (e.g., as a result of training).

The question that needs to be answered is: When a new document d arrives, what would be the best assignment of d to the concept set based on the knowledge in matrices A and B? The answer can be obtained from a solution to the Linear Least Square Fit (LLSF) problem as to be discussed below (Yang and Chute, 1994) and SVD can be used to obtain such a solution.

The LLSF problem is to find a $k \times n$ mapping matrix F that minimizes the following sum of residual squares (Yang and Chute, 1994):

$$\sum_{i=1}^{m} \| \vec{e}_i \|_2^2 = \sum_{i=1}^{m} \| F\vec{a}_i^T - \vec{b}_i^T \|_2^2 = \sum_{i=1}^{m} \| FA^T - B^T \|_F^2 \tag{1}$$

where matrices A_{m*n} and B_{m*k} are given above; A^T and B^T are their transposes; $\vec{a}_i$ and $\vec{b}_i$ are the ith rows in A and B, respectively; $\vec{e}_i = F\vec{a}_i^T - \vec{b}_i^T$ is the error of F in the assignment of $\vec{a}_i$ to $\vec{b}_i$; $\| \cdots \|_2 = \sqrt{\sum_{j=1}^{k} x_j^2}$ is the vector 2-norm of a vector and $\| \cdots \|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{l} y_{ij}^2}$ is the Frobenius matrix norm of a matrix. The LLSF problem has at least one solution. A conventional method for solving the LLSF problem is to use singular value decomposition (Golub and Van Loan, 1989; Lawson and Hanson, 1974). The solution is:

$$F = B^T(A^+)^T = B^T U S^{-1} V^T \tag{2}$$

where A^+ is the pseudoinverse of matrix A and $A^+ = VS^{-1}U^T; U_{m*p}, V_{n*p}$, and S_{p*p} are matrices obtained by the SVD in that $A = USV^T$; U and V contain the left and right singular vectors, respectively; $S = diag(s_1, \ldots, s_p)$ contains p nonzero singular values satisfying $s_1 \geq \cdots \geq s_p > 0, p = \min(m, n, k)$; $S^{-1} = diag(1/s_1, \ldots, 1/s_p)$ is the inverse of S. Matrix F is also called a term-category association matrix and element f_{ij} in F represents a score/weight that the jth term is related to the ith category, $i = 1, \ldots, k$ and $j = 1, \ldots, n$.

With the above discussion, the process of assigning a new document d (or its vector format $\vec{d}$) to categories can be summarized as follows. First, transform $\vec{d}$ to a new vector $\vec{d^*}$ in the category space: $\vec{d^*} = (F\vec{d}^T)^T$. Second, apply the Cosine function to compute the similarity between $\vec{d^*}$ and each category vector $\vec{c}$. The vector of a category contains only one nonzero element (i.e., 1) corresponding to the category in the category space. Finally, $\vec{d^*}$ is assigned to the k categories with the highest similarities for some integer k (see Section 3.1 on the discussion of this parameter).

The above document categorization algorithm can be applied to categorize databases as follows. First, we will represent each database as a document through its centroid (see Section 3.1). From the centroids of all databases, the matrix A mentioned above can be created in which each row corresponds to a centroid of a database and each column corresponds to a term. Matrix B can be created based on manually assigning a number of databases to the set of concepts in the concept hierarchy. From matrices A and B, the term-category association matrix F can be found using SVD. When assigning a new database, we first compute the centroid of the database and then transform it to a new vector using F. After this, the process of assigning the database to concepts is the same as that for assigning a transformed document.

The performance of this method is sensitive to the selection of the training data. If the vocabulary of the training data covers well the vocabulary of the documents (databases) to be assigned, then this method can perform well.

Otherwise, the performance may suffer. As a result, carrying out high quality training can be difficult in practice as the selection of good training data requires the trainer to have a good idea about the real data. Training is also a very time-consuming task. Finally, applying the SVD method for database categorization also suffers the same problem as the HSDC method due to the use of database centroids.

3.3. High Average Similarity over Retrieved Documents (HASRD)

In this strategy, we treat each concept description as a query. Such a query can be represented as a vector of terms with weights. The query vector is the same as the vector of the description (see Section 3.1). This strategy works as follows.

1. Calculate the similarity for each concept and database pair (q, D), where q is a concept query and D is a database. This is accomplished as follows.
   (a) Submit q to the search engine of database D.
   (b) Retrieve the M documents from D that have the largest similarities with q for some integer M. First calculate the similarities of these documents with q and then calculate the average of these similarities. This average similarity will be treated as the similarity between the query (concept) q and the database D.
2. For each given database D, sort all concepts in non-ascending similarities between all concepts and the database and then assign D to the top k concepts for some integer k (same as the k in Section 3.1).

We can carry out experiments to determine an appropriate value for parameter M (see Section 4).

This strategy also has the following features.

1. It is easy to apply in practice. This is because this strategy does not require accessing all the documents in a database. It only needs to submit each concept as a query to each database and analyze a small number of returned documents from each database. In addition, this method does not require cooperation from local search engines.
2. It is unique for categorizing databases. In other words, this strategy is not applicable for categorizing documents.

4. Experiments

In this section, we report our preliminary experimental results.

4.1. Test Data

Twenty-four document databases are used in our experiment. Among them, 18 are snapshots of newsgroups randomly chosen from a list downloaded from ftp://rtfm.mit.edu/pub/usenet-by-group and 6 are web page collections fetched from 6 web sites in two US universities. The number of documents in these

dbID	Homepage URL/newsgroup names	#doc	#concepts
1	www.cs.binghamton.edu	1024	13
2	www.ee.binghamton.edu	314	21
3	www.arthist.binghamton.edu	28	26
4	www.athletics.binghamton.edu	488	20
5	www.law.harvard.edu	872	22
6	www.medical.harvard.edu	1632	27
102	alt.fan.jai-maharaj	223	20
122	alt.folklore.herbs	15	12
136	alt.happy.birthday.to.me	253	12
152	alt.internet.research	27	16
169	alt.magick.tyagi	14	16
263	alt.religion.islam	17	13
325	alt.support.big-folks	12	11
329	alt.support.depression	12	13
439	comp.benchmarks	16	16
480	comp.lang.c++	16	16
509	comp.mail.misc	18	10
613	comp.sys.sgi.misc	12	11
626	comp.unix.questions	13	14
717	misc.consumers.frugal-living	16	14
733	misc.invest.options	16	15
734	misc.invest.stocks	16	15
735	misc.invest.technical	16	13
742	misc.kids.info	44	15

Table 1. 24 Document Databases Used in the Experiments

databases varies from 12 to 1,632 (see Table 1 for more information about the test databases).

As discussed in Section 3, we created a concept hierarchy based on the first two levels of the Yahoo category hierarchy. The first level has 12 concepts while the second level has 352 concepts. We manually assigned each of the 24 databases to related concepts in the concept hierarchy. The result of the manual assignment serves as the baseline (i.e., ideal performance) for evaluating different database categorization methods. An ideal method should achieve the same performance as the manual assignment. On the average, each database is assigned to close to 16 concepts. The last column in Table 1 indicates, for a given database, the number of concepts that the database is manually assigned to.

The performance measures of each database categorization method are given as follows.

1. **recall**: the ratio of the number of correctly assigned concepts over the number of all correct concepts. For example, suppose a given database has 10 correct concepts (i.e., the database should be assigned to 10 concepts if the assignment is perfect) and a given database categorization method assigns the database to 4 of the 10 concepts. Then the recall of this method for assigning this database is 40% or 0.4. The average of the recalls of a given database categorization method over all test databases is called the recall of this method.
2. **precision**: the ratio of the number of correctly assigned concepts over the number of all assigned concepts. For example, suppose a given database is assigned to 10 concepts by a database categorization method and among the 10 concepts 6 are assigned correctly. Then the precision of this method for assigning this database is 60% or 0.6. The average of the precisions of a given database categorization method over all test databases is called the precision of this method.

The two quantities, recall and precision, together measure the performance or effectiveness of a database categorization method. If a method achieves the performance with recall = 1 and precision = 1, that is, the method assigns each database to exactly the set of correct concepts, no more and no less, then the performance of the method is perfect. Perfect performance is unlikely to be achieved in practice. Usually when recall increases, precision goes down.

4.2. Experiments and the Results

To compare the three database categorization strategies discussed in Section 3, we decide to carry out experiments to draw the 11-point average recall-precision curve for each strategy. For each strategy, the average precision at each of the 11 recall points 0, 0.1, 0.2, ..., 1.0 is obtained and the curve is then drawn based on the 11 recall-precision points. For each strategy, in order to achieve a desired recall, a proper value of the parameter k is chosen for each database (smaller k leads to a lower recall and larger k leads to a higher recall). For strategy HSDC, the experiments can be carried out directly as the only parameter involved is k. For strategy HASRD, in addition to the parameter k, another parameter M is involved. After testing with different values of M, we observed that when $M = 10$, good performance for the HASRD strategy can be obtained. Therefore, during the experiments to draw the 11-point average recall-precision curve for strategy HASRD, $M = 10$ is used.

To test the Singular Value Decomposition (SVD) strategy, we need to have a training set so that matrices A and B can be obtained. To do this, we select 20 databases from our 24 databases as training databases to build the matrices A and B, and use the remaining 4 databases as test databases to evaluate the performance of the SVD strategy. Matrix A has 20 rows, corresponding to the centroids of the 20 training databases, and 40,671 columns, corresponding to the distinct terms in all databases. For the 20 training databases, we manually assign each database to the concepts in our concept hierarchy. Matrix B is a 20*364 matrix with 364 corresponding to the 364 concepts in both levels of the concept hierarchy. If a training database is assigned to a concept, then the corresponding entry in B has a value of 1; otherwise, the corresponding entry is zero.

Training Set Id	Database IDs
1	All except 2, 325, 480, 735
2	All except 122, 152, 613, 733
3	All except 1, 329, 439, 734

Table 2. The Three Training Sets for Testing the SVD Method

As we mentioned in Section 3.2, the performance of the SVD method is sensitive to the completeness of the training set. In database categorization, if the training databases used to build matrices A and B contain all the terms, including those that may appear in the test databases that will be assigned to the concept hierarchy, then the test result will likely be much better. Based on this observation, the 20 training databases are chosen such that they will contain all or almost all of the terms in the remaining test databases. In order to avoid/reduce the inaccuracy of the experiments based on the choice of a single set of training databases, three different sets are chosen and used in our experiments. The three training sets of databases are listed in Table 2 (Table 2 actually lists the databases not used in each training set). For each training set, the remaining 4 databases are used for testing. Note that *cross validation* is a widely used technique for testing the performance of a method, say O. With this technique, the test data are partitioned into N equal-sized subsets for some integer N and $(N > 1)$ tests are carried out. In each test, data from all but one subset are used for training and the data from the remaining subset are used for validation (performance measure). The average performance of the $(N > 1)$ tests is used as the final performance of the method O. The validation technique we use here can be considered as a modified cross validation in the sense that the databases in the training set are required to contain all or almost all terms in the corresponding test set.

For the SVD strategy, the average recalls and precisions are based on the three training sets. The three 11-point average recall-precision curves for the three database categorization strategies are shown in Figure 1. It can be seen that the HASRD method is consistently better than the other two methods, often by substantial margins, especially for high recalls. The SVD method performed better than the HSDC method when the recalls are small but the opposite is true when the recalls are large.

The overall performance of a method can be measured by the 11-point average precision, which is computed by averaging the 11 precision values of the method for the 11 recall points. The 11-point average precision is 0.795 for method HASRD, 0.724 for method HSDC, and 0.688 for method SVD. Based on the 11-point average precision, method HASRD is about 10% better than method HSDC and about 16% better than the SVD method. For recalls between 0 and 0.3, both the HASRD method and the SVD method have perfect precisions. But for recalls between 0.4 and 1, the (7-point) average precision is 0.679 for method HASRD and 0.51 for method SVD, meaning that on the average method HASRD performed 33% better than method SVD for these recalls.

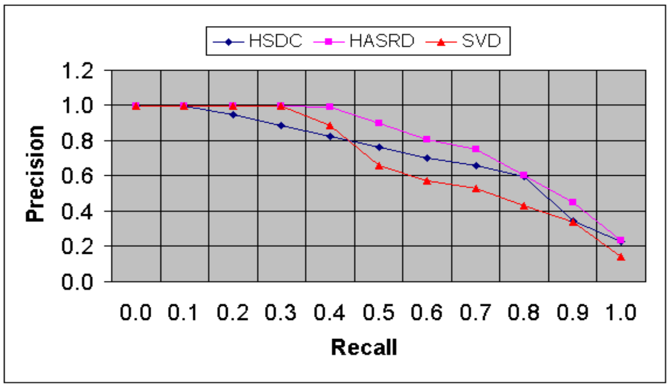

Fig. 1. 11-point Recall-Precision Curves of the Three Methods

5. Implementation of a Prototype System

In this section we describe a prototype database categorization system. The objective is to develop a practical system that can categorize any given search engine when the URL of the search engine is provided. The HASRD algorithm is chosen for the prototype system due to its good categorization performance and its easiness to implement.

In order to categorize any search engine with a given URL, a program is developed to automatically connect to any given search engine and to pass queries to it. This is achieved by analyzing the source HTML file of the search engine page. From the source file, the HTTP request method (GET or POST) supported by the search engine can be identified from the *method* attribute of the *form* tag; the CGI (Common Gateway Interface) program of the search engine server can be identified from the *action* attribute of the *form* tag; and the query name can be identified from the *name* attribute of the *input* tag. The GET method is often used to send a simple request to a server with query terms as parameters following a question mark (?). The POST method permits more substantial amount of data to be passed to a server. In this case, the data is passed to the server in the entity body of the request. Both GET and POST are widely used by search engines. Among the 190 commercial search engines listed at site www.searchenginewatch.com that we used to test our program, 128 support the GET method and 49 support the POST method (13 are unknown). The CGI program is where each request should be sent. This program needs to be identified in a request. The query name lets the search engine server identify the query terms in a request. After the above pieces of information (i.e., HTTP request method, CGI program and query name) are obtained, writing the code (in Java) is reasonably easy. We tested our connection program using the above 190 commercial search engines and a success rate of 93% was achieved. Next, another program is developed to identify the URLs of the retrieved documents from the result page of a search engine with respect to any given query. These URLs are needed to actually download the retrieved documents so that their similarities with the query can be computed. Note that for many commercial search engines, a result page often contains URLs of advertisement pages and other unrelated pages in addition to the URLs of the retrieved documents for a given query.

Therefore, this program needs to separate desired URLs from unwanted URLs. In addition, the program should work for the result page of each search engine. It is non-trivial to develop a high quality URL-filtering program. We will not discuss the URL-filtering algorithm used in our program in this paper as the algorithm is not directly related to database categorization.

The concept hierarchy used in our prototype system is the Open Directory hierarchy, which can be obtained from a number of search engine sites including the Google site (www.google.com), rather than the Yahoo hierarchy. The main reason of our (change of) choice is that the Open Directory hierarchy is more widely used and of a better quality than the Yahoo hierarchy. The first two levels of the Open Hierarchy are used. There are 15 concepts in the first level and 603 concepts in the second level.

The database categorization algorithm HASRD is modified slightly for the prototype system. The main change is that the algorithm no longer automatically assigns a database to the parent concept when the database is assigned to a child concept. Instead, an assignment propagation scheme is employed to determine whether or not a database should be assigned to the parent concept. This scheme is described below. First, when a search engine S is submitted to the system for categorization, each of the descriptions of the 603 concepts in the second level of the concept hierarchy is sent to the search engine as a query. The average similarity of the top k returned documents for each query is computed. If the average similarity is above a user-specified threshold T, then the search engine S is assigned to the concept. If the user does not provide the values for k and T, then default values will be used. Next, for the given search engine S and each concept C in the first level, the percentage of the child concepts of C to which S is assigned is computed. If the percentage exceeds a user-specified percentage threshold P, then S is assigned to C; otherwise, S is not assigned to C. Again, if the user does not provide a value for P, then a default value will be used. When S is assigned to C, the average of the similarities of the child concepts of C with S is computed and is used as the similarity between S and C. If a search engine is assigned to P or higher percentage of the concepts in the first level of the concept hierarchy, then the search engine is considered to be assigned to the dummy root of the hierarchy and is declared as a general-purpose search engine. In the experimental results presented below, $k = 10$, $T = 0.05$ and $P = 0.8$ (i.e., 80%) are used. These values are currently determined based on our experience with this prototype system.

The above discussions are summarized into the following database categorization algorithm. In the algorithm, query(c) denotes the description of concept c in the concept hierarchy, L_1 and L_2 denotes the set of concepts in the first level and the second level, respectively.

Algorithm DBCategorization(S, H, k, T, P)
/* S is the URL of a search engine; H is the concept hierarchy; */
/* k is the # of top ranked documents used to compute the avg. similarity; */
/* T is the similarity threshold for assigning S to a concept in L_1; */
/* P is the % threshold for assigning S to a concept in L_2 and the root. */
begin
  for each concept c in L_1
    { submit query(c) to S and fetch the top k returned documents;
     compute the average similarity avg_sim of the k documents with query(c);
     if ($avg\_sim > T$)

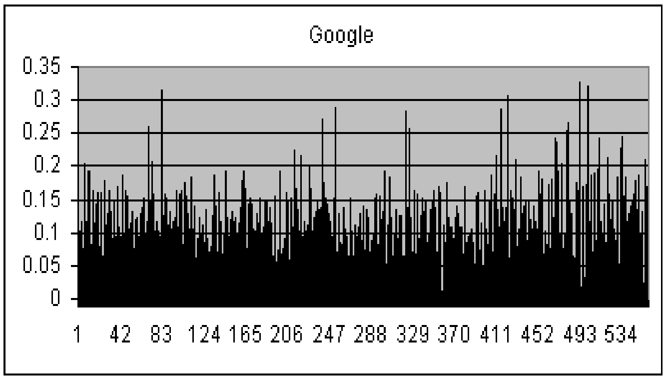

Fig. 2. Average Similarity Distribution Based on the Second Level Concepts for Google

```
        assign S to c;
  }
  for each concept c in L2
    { count the number N of child concepts to which S is assigned;
      if (N > P)
        { assign S to c;
          compute the average of the similarities of the child concepts of c;
        }
  }
  count the number M of concepts in L2 to which S is assigned;
  if (M > P)
    { assign S to the root;
      print("This is a general-purpose search engine.");
      exit;
    }
  for each concept c in L2
    if S is assigned to c
      print c and the average similarity;
  for each concept c in L1
    if S is assigned to c but not to the parent of c
      print c and the average similarity;
end;
```

Figure 2 shows the distribution of the average similarities of the Google search engine with respect to all concepts in the second level. In Figure 2, the X-axis lists all the 603 concepts (not all of their numbers are shown) in the second level and the Y-axis indicates the average similarities of different concepts. It can be seen that when the description of almost every concept is submitted as a query to the Google search engine, decent result is returned.

Figure 3 shows the average similarity distribution when the search engine assignments are propagated to the concepts in the first level. It is easy to see that the similarity distribution is quite uniform across all the 15 concepts. The Google search engine is recognized as a general-purpose search engine by our database categorization system.

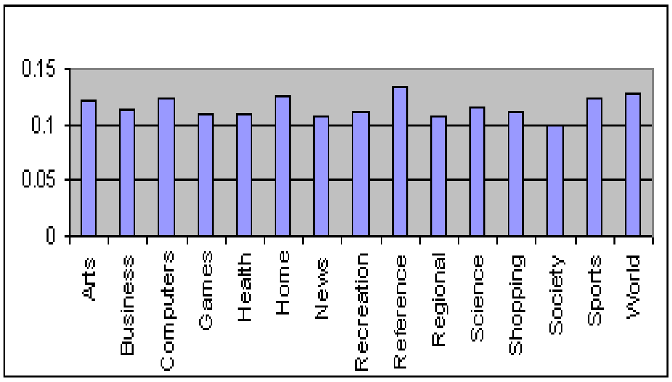

Fig. 3. Average Similarity Distribution Based on the First Level Concepts for Google

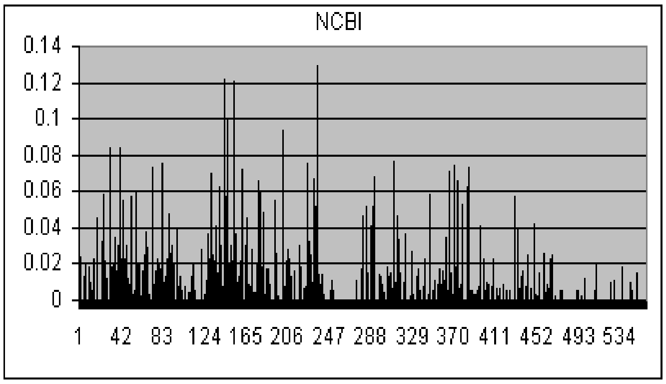

Fig. 4. Average Similarity Distribution Based on the Second Level Concepts for NCBI

Figure 4 shows the distribution of the average similarities of the NCBI (National Center for Biotechnology Information) search engine (www.ncbi.nlm.nih.gov) with respect to all concepts in the second level. It can be seen that the similarity distribution is very uneven. It retrieved good results for a small number of concepts while returned no or very poor results for most concepts. Figure 5 shows the average similarity distribution when the search engine assignments are propagated to the concepts in the first level. It is clear from Figure 5 that this search engine is a specialized search engine covering primarily areas in health and reference.

6. Conclusions

Database categorization can help database selection algorithms in a metasearch engine choose more relevant databases to search for a given user query. This can lead to higher retrieval effectives of the metasearch engine. In this paper, we studied three database categorization algorithms. Two of the three studied methods (HSDC and SVD) can be considered as adoptions of document categorization techniques to the database categorization problem and the other

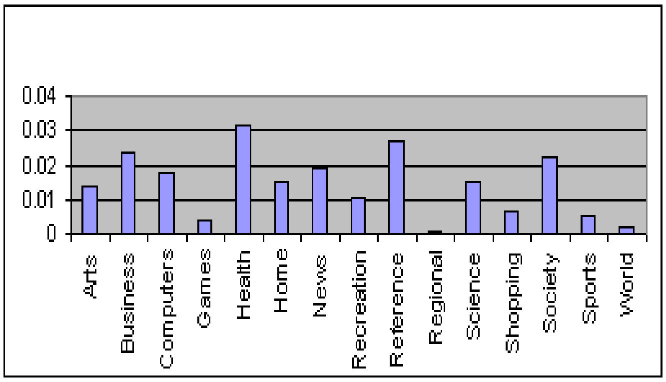

Fig. 5. Average Similarity Distribution Based on the First Level Concepts for NCBI

method (HASRD) is only applicable to the database categorization problem. HASRD is simple and easy to apply as it does not need statistical information about the documents in a database (as HSDC does) nor does it need training (as SVD does). Furthermore, based on our experiments, HASRD has the best overall performance among the three methods studied. This indicates that simple categorization algorithms that take into account the special characteristics of databases can outperform sophisticated algorithms adopted from document categorization. Based on the HASRD algorithm, a database categorization prototype system has been implemented. Preliminary test based on real Web search engines indicates that the system can produce good result.

We plan to carry out the following research in the near future. First, we plan to expand our testbed by including more databases. This will enable us to carry out additional experiments to obtain more accurate experimental results. Second, we plan to improve the HASRD algorithm and develop new database categorization algorithms. Additional information such as the number of documents (hits) retrieved by a search engine for a query may also be useful for database categorization and we would like to study how to incorporate the information into the algorithm. Third, the HASRD algorithm as presented in Section 5 has a number of parameters (i.e., k, T, and P) and we would like to determine the best values for these parameters. Fourth, we would like to improve the efficiency of the HASRD algorithm. At present, 2-3 hours are needed to categorize a single search engine as over 600 queries need to be submitted and processed. Another interesting issue to investigate is to determine an appropriate number of levels in the concept hierarchy. Using too few levels of the concept hierarchy could mean over-generalization while using too many levels could mean over-specialization. Both extremes may lead to less effective retrieval.

Acknowledgements. We thank anonymous reviewers for their very valuable comments and suggestions. This work is supported in part by the following NSF grants: IIS-9902792, IIS-9902872 and EIA-9911099.

References

Baumgarten C (1997) A probabilistic model for distributed information retrieval. Proceedings of ACM SIGIR Conference, Philadelphia, PA, July 1997, pp. 258-266.

Callan J, Lu Z, Croft W (1995) Searching distributed collections with inference networks. Proceedings of ACM SIGIR Conference, Seattle, 1995, pp. 21-28.

Dreilinger D, Howe A (1997) Experiences with selecting search engines using metasearch. ACM TOIS, July 1997, 15(3): 195-222.

Fan Y, Gauch S (1999) Adaptive agents for information gathering from multiple, distributed information sources. Proceedings of 1999 AAAI Symposium on Intelligent Agents in Cyberspace, Stanford University, March 1999, pp. 40-46.

Furnas G, Deerwester S, Dumais S et al (1988) Information Retrieval Using a Singular Value Decomposition Model of Latent Semantic Structure. Proceedings of ACM SIGIR Conference, Grenoble, France, June 1988, pp. 465-480.

Golub G, Van Loan C (1989) Matrix Computations. 2nd Edition, The John Hopkins University Press, 1989.

Gravano L, Garcia-Molina H (1995) Generalizing GlOSS to vector-space databases and broker hierarchies. Proceedings of the International Conferences on Very Large Data Bases, Zurich, Switzerland, September 1995, pp. 78-89.

Jansen B, Spink A, Bateman J, Saracevic T (1998) Real life information retrieval: A study of user queries on the Web. ACM SIGIR Forum, 32(1).

Kahle B, Medlar A (1991) An Information System for Corporate Users: Wide Area Information Servers. Technical Report TMC199, Thinking Machine Corporation, April 1991.

Kirsch S (1998) The future of Internet search: Infoseek's experiences searching the Internet. ACM SIGIR Forum, 32(2): 3-7.

Kleinberg J (1998) Authoritative sources in hyperlinked environment. Proceedings of ACM-SIAM Symposium on Discrete Algorithms, San Francisco, California, January 1998, pp. 668-677.

Koster M (1994) ALIWEB: Archie-Like indexing in the Web. Computer Networks and ISDN Systems, 27(2): 175-182.

Lawson C, Hanson R (1974) Solving least squares problems. Prentice-Hall, 1974.

Liu K, Yu C, Meng W et al (2001) A statistical method for estimating the usefulness of text databases. IEEE Transactions on Knowledge and Data Engineering (to appear).

Manber U, Bigot P (1997) The search broker. Proceedings of the USENIX Symposium on Internet Technologies and Systems, Monterey, California, 1997, pp. 231-239.

Meng W, Liu K, Yu C et al (1998) Determine text databases to search in the Internet. Proceedings of the International Conference on Very Large Data Bases, New York City, August 1998, pp. 14-25.

Meng W, Liu K, Yu C et al (1999) Estimating the usefulness of search engines. Proceedings of IEEE International Conference on Data Engineering, Sydney, Australia, March 1999, pp. 146-153.

Page L, Brin S, Motwani R, Winograd T (1998) The PageRank citation ranking: Bring order to the Web. Technical Report, Stanford University, 1998.

Salton G, McGill M (1983) Introduction to modern information retrieval. New York: McCraw-Hill, 1983.

Selberg E, Etzioni O (1997) The MetaCrawler architecture for resource aggregation on the Web. IEEE Expert, 12(1): 8-14.

Wang K, Zhou S, Liew SC (1999) Building hierarchical classifiers using class proximity. Proceedings of the Very Large Data Bases Conference, September 1999, Edinburgh, Scotland, UK., pp. 363-374.

Xu J, Callan J (1998) Effective retrieval with distributed collections. Proceedings of the ACM SIGIR Conference, Melbourne, Australia, 1998, pp. 112-120.

Xu J, Croft B (1999) Cluster-based language models for distributed retrieval. Proceedings of the ACM SIGIR Conference, Berkeley, California, August 1999, pp. 254-261.

Yang Y, Chute C (1994) An example-based mapping method for text categorization and retrieval. ACM Transactions on Information Systems, July 1994, 12(3): 252-277.

Yu C, Liu K, Wu W et al (1999) Finding the most similar documents across multiple text databases. Proceedings of the IEEE Conference on Advances in Digital Libraries, Baltimore, Maryland, May 1999, pp. 150-162.

Yu C, Liu K, Meng W et al (2001) A methodology to retrieve text documents from multiple databases. IEEE Transactions on Knowledge and Data Engineering (to appear).

Yu C, Meng W, Liu K et al (2000) Efficient and effective metasearch for text databases incorporating linkages among documents. Proceedings of the ACM SIGMOD Conference, Santa Barbara, CA, May 2001.

Yuwono B, Lee D (1997) Server ranking for distributed text resource systems on the Internet. Proceedings of the fifth International Conference on Database Systems For Advanced Applications, Melbourne, Australia, April 1997, pp.391-400.

Author Biographies

meng.jpg

Weiyi Meng received the BS degree in mathematics from Sichuan University, People's Republic of China, in 1982, and the MS and PhD degrees in computer science from the University of Illinois at Chicago in 1988 and 1992, respectively. He is currently an associate professor in the Department of Computer Science at the State University of New York at Binghamton. His research interests include Web-based information retrieval, multidatabase systems, query processing and optimization. He is the co-author of a database book (Principles of Database Query Processing for Advanced Applications). He has published more than 50 papers in various journals and international conference proceedings. He has served/is serving as program chair and program committee member of many international conferences.

wang.gif

Wenxian Wang received her MS and BS degrees in electric engineering from Tsinghua University, Beijing, China, in 1994 and 1991, respectively. She received her MS degree in computer science from State University of New York at Binghamton in January 2000. She currently works as a Programmer Analyst in SBS International of New York Inc., participating in Multi-tier Client/Server system design and development.

sun.jpg

Hongyu Sun received his BS degree from Beijing University, P. R. China in 1995. He received his MS degree in computer science from the State University of New York at Binghamton in 2000. He currently works as a Software Engineer in Open Channel Solutions, a Boston based company. He is interested in Java Programming and Database System.

yu.jpg

Clement Yu is a professor in the Department of Computer Science at the University of Illinois at Chicago. He obtained his B.S. degree in applied mathematics from Columbia University in 1970 and his Ph.D. degree in Computer Science from Cornell University in 1973. His areas of interest include search engines and multimedia retrieval. He has publications in various journals such as IEEE TKDE, ACM TODS and JACM and in various conferences such as VLDB, ACM SIGMOD, ACM SIGIR and ACM Multimedia. He previously served as chairman of ACM SIGIR and as a member of the advisory committee to the National Science Foundation. He was/is a member of the editorial board of IEEE TKDE, International Journal of Software Engineering and Knowledge Engineering and Distributed and Parallel Databases. He was a co-chair of the National Science Foundation Information and Data Management Program Workshop for principal investigators for the year 2000 and a co-chair of the International Conference on Information Society in the 21st Century: Emerging Technologies and New Challenges which was held in Japan.

Correspondence and offprint requests to: Weiyi Meng, Department of Computer Science, State University of New York at Binghamton, Binghamton, NY 13902, USA. Email: meng@cs.binghamton.edu