

# Web Based Collection Selection Using Singular Value Decomposition

John King and Yuefeng Li  
School of Software Engineering and Data Communications  
Queensland University of Technology  
QLD 4001, Australia  
{j5.king,y2.li}@qut.edu.au

## Abstract

*As the number of electronic data collections available on the internet increases, so does the difficulty of finding the right collection for a given query. Often the first time user will be overwhelmed by the array of options available, and will waste time hunting through pages of collection names, followed by time reading results pages after doing an ad-hoc search. Automatic collection selection methods try to solve this problem by suggesting the best subset of collections to search based on a query. This is of importance to fields containing large number of electronic collections which undergo frequent change, and collections that cannot be fully indexed using traditional methods such as spiders. This paper presents a solution to this problems of selecting the best collections and reducing the number of collections needing to be searched. Preliminary tests of the system, conducted on Web search engines, suggest that this will solve much of the Web based Collection selection problem.*

## 1 Introduction

*Collection selection* is the selection of an optimal subset of collections from a large set of collections for the purpose of reducing costs associated with Distributed Information Retrieval [4, 11]. The goal of collection selection is to make searching multiple collections appear as seamless as searching a single collection [18]. Another requirement of a collection selection system is to learn which collections contain relevant information and which collections contain no relevant information. This reduces the number of overall search requests needed. If only a small high quality subset of the available collections are searched then savings can be made in time, bandwidth, and computation.

Web based collection selection is significant because as the internet grows the number of internet based collections grows. It is now impossible to manually track and index all collections as they number in the thousands. This re-

search will enable users to choose the best collections for their needs without having to sift through irrelevant collections. The need for this research will keep growing as more and more information resources are made available electronically.

There are a number of differences between traditional and Web based collection selection. Web based collections typically return a set of ordered results from a query, whereas traditional collection commonly return unordered results. It is difficult to estimate the size and density of Web based collections because most Web based collections do not provide a way of finding the frequency of terms and the number of available documents. This also means that commonly used metrics such as recall cannot be used with Web based collections. Many collections on the Web are not indexable by traditional indexing methods as they only present a search interface and no way of indexing the contents. Many Web based collections also change frequently, meaning that traditional indexing methods have to be run frequently in order to keep index information valid. Web based collection selection is typically performed using a *meta(or multi) search engine*, a program that queries a number of Web based collections and combines the results into a single result list ordered by relevancy. Meta search engines commonly use collection selection to improve results and cut down bandwidth usage.

The difficult collection selection problems are reducing expenses, increasing search speed, learning to adapt to change in the search environment, using ontology to increase precision/recall, and learning to adapt to the users preferences. This paper introduces solutions to the evaluation and ranking of collections, and the using of feedback to select the best related collections for each search. The solutions use a novel application of singular value decomposition in order to find relationships between samples of collections, and to use user precision feedback to find user preferred collections. The solution does not require large amounts of meta-data about each server, and does not require any of the servers to keep meta-data about themselves.

Section 2 describes Collection Selection. Section 3 introduces Latent Semantic Analysis. Section 4 outlines term and collection relations. Section 5 outlines precision and collection relations. Section 6 describes how the system will be evaluated, and finally Section 7 concludes the paper.

## 2 Collection Selection

Collection Selection(or resource discovery) is the selection of an optimal set of information sources from a large set of information sources. An information source can be a Web interface, a standard relational collection, a file, a search engine, or any other textual representation of information. Collection Selection aims to be efficient with respect to bandwidth and computation, and decreases both resource usage and time taken to return a set of results for a query. Well planned collection selection can have a large influence on the efficiency of a query. Lu et al [16] and Choi et al [5] state that there is a high correlation between the search performance and the distribution of queries among collections.

There are two main approaches to collection selection. The first is to generate a full index of every term in every available collection. The second approach involves taking samples of each collection. Each approach has its benefits and drawbacks.

Creating a full index of every collection is the more accurate(and expensive) approach. In the case of the Web full indexing would mean that every one of the thousands of available collections would be fully crawled and indexed. With common indexing techniques, such as inverted indexes, the size of this index is approximately half the size of the original collection. This crawl also would have to be periodically scheduled, to keep newer material available. Creating these indexes consume large amounts of computation, bandwidth, and memory, which means that this approach is available only to a few.

Because of the size and rate of change of many collections, it is often difficult and expensive to create an index of each collection used, and thus small samples of the highest ranked (or most representative) documents can be taken from each collection instead, and these samples are treated as being representative of the entire collection. This would mean that a short query could be broadcasted to the entire set of collections, and top results these collections returned could then be ranked in order of relevance to the query. This sampling technique works well with large sets of collections. Menczer et al [17] notes that in cases where ranking of the entire collection is not possible, standard performance measure such as precision recall curves are of little use, so other metrics need to be used. In this paper a method of finding related collections will be presented to reduce the overhead of searches and the number of collections need-

ing to be searched each time. A way of clustering related collections using user feedback will also be presented.

### 2.1 Related Work

We will now discuss CORI and GLOSS, two of the most effective and popular collection selection techniques.

CORI(*Collection Retrieval Inference Network*) is a Bayesian probabilistic inference network which is commonly used for document selection. Callan [4] and later Lu [15] apply CORI to the collection selection problem. Callan's solution is elegant, and is a popular collection selection method.

CORI assumes that the best collections are the ones that contain the most documents related to the query. These collections are therefore ranked in order of number of documents relevant to the information needed. Once ranked the top  $n$  collections are presented to the user in a list. Methods are available for calculating the number of documents about a particular term in a collection, and for calculating normalised document frequency, inverse collection frequency, and the importance of a collection for a particular term.

Gravano's [13] solution to the collection selection problem is use a server which contains all the relevant information of other collections. Users query this *Glossary of Servers Server* (or GLOSS for short) which then returns a ordered list of the best servers to contact to send the query to.

GLOSS is used to evaluate and rank collections. Collections are evaluated by the usefulness for each query. This usefulness is measured by the number of documents in the collection that are similar to the query. The collections most likely to be selected contain many documents relevant to the current query. Collections are ranked based on information about each collection. However full data on each collection cannot be stored due to size restrictions. Instead each word and each collection is given a number based on the weight of the word and the number of documents in the collection that contain the word. These numbers are periodically updated by a collector program which gets this information from each collection. GLOSS works best with a large collection of heterogeneous data sources.

In a comparison of CORI and GLOSS [7] it was found that CORI was the best selected method, and that a selection of a small number of collections could outperform selecting all the servers and a central index. Probe queries are a good method for evaluating collections without having full knowledge of the collection contents, with 50,000 documents evaluated instead of 250,000 documents.

These conclusions are important to our research because it shows that a high quality subset of collections will be as effective as a full set of collections, and that probes of a collection are an effective method of ranking an entire col-

lection.

## 2.2 Collection Selection vs Document Selection

Collection selection is significantly different to document selection in a number of areas. Collection selection uses different methods to document selection for scoring an items relevance. Document selection commonly uses a binary relevance value, which collection selection cannot use. Instead collection selection must use a floating point number to represent relevance. Collection selection also differs from document selection in that it uses different ways of calculating term weighting. (terms distributed across all documents in a collection are worth more than terms clustered in one document of a collection) Another difference between collection selection and document selection is that different content selection methods are needed, with Web based collection selection commonly using partial collection sampling, and document selection using full document indexing. These differences mean that collection selection requires a significantly different approach to document selection.

## 2.3 Collection Selection Metrics

Collection selection uses different precision and recall measurements to document selection. This collection selection metric set is an extension of the document precision and recall metrics, first mentioned in Gravano *et al* [12]. Collection Precision is the percentage of relevant collections in relation to the number of collections retrieved. Collection Recall compares the amount of relevance in all the collections with the relevance in the top n collections for a query.

This research will use the collection precision and recall formulae presented in Callans [3] paper. In the case of Web based Information Retrieval systems relative recall is used instead of recall. The following two formulae are used for calculating collection precision and collection recall.

$$Precision_n = \frac{\sum_{i=1}^n \begin{cases} 1 & \text{if } NumRel(db_{ei}) > 0 \\ 0 & \text{otherwise} \end{cases}}{n}$$

$$Recall_n = \frac{\sum_{i=1}^n NumRel(db_{ei})}{\sum_{i=1}^n NumRel(db_{bi})}$$

where

- $e$  is estimated collection ranking
- $b$  is the corresponding relevance based collection ranking (*baseline*)
- $n$  is the rank

- $NumRel(db_{ji})$  is the number of relevant documents in the  $i$ th collection of collection ranking  $j$
- $db$  is the collection

These measurements will be used to judge the collection selection methods used in this research, and are a central part of the methodology used to compare this work with other algorithms.

## 3 Latent Semantic Analysis

Latent Semantic Analysis(LSA) is a vector space model for analysing relationships within a matrix. Developed by Deerwester [8] and Berry [1], it is a statistical method of reducing the noise in a matrix and associating related concepts together. Used with a term-document matrix, Latent Semantic Analysis takes the background structure of word usage and removes the noise. This noise reduction allows the higher order relationship between terms and documents to be clearly seen [8].

A big problem with traditional information retrieval methods is that a user will often have a general concept of what they are searching for, but will not know the exact term to describe the concept, or that the term they are using has many different meanings. Deerwester [8] refers to these problems as *polysemy* and *synonymy*. *Polysemy* is a word having multiple meanings. An example of polysemy is that a student may be looking for information about a “jaguar” car, and so enters the term “jaguar”. However this term can also mean jaguar “cat”, and the Mac OS X “Jaguar” operating system. In computational linguistics, this is called word sense disambiguation. Term matching, while a partial solution to this problem, returns irrelevant documents and this reduces precision. *Synonymy* is multiple words having the same meaning. For example the term “cat” can be also referred to as “feline”, “lion”, “kitten”, and so on. Synonymy can be partially solved by human generated thesaurus, however this causes the problem of inconsistent human judgments occurring in human generated indexes. Based on context, the human reader has little problem inferring meaning about these words, however it is difficult for machines to infer meaning about these words. Latent Semantic Analysis helps with these problems, in fact the term itself does not have to occur within the collection for Latent Semantic Analysis to find that the collection is relevant. This is a great feature with off-the-page rankings being so popular with search engines at present. With Latent Semantic Analysis it is possible to fingerprint each collection and measure a sample of each collection against a query.

Chen *et al* [6] proved that Latent Semantic Analysis consistently improves both recall and precision. The Latent Semantic Analysis method has equaled or outperformed

standard vector retrieval methods in almost every case, and gives up to 30% better retrieval [9].

A popular use of LSA is to analyse document/document relationships, however this research uses it to analyse collection/collection relationships. LSA is very suitable for collection selection, more so than document collection, because of its ability to match related terms. LSA is not suitable for some document selection tasks (such as religious document search) where only an exact match is useful, however it is this property that makes LSA so useful across collections where an exact match of a work is not required, only a match in concept is required.

Dimensionality reduction is another important part of Latent Semantic Analysis, allowing 10,000 dimensions to be reduced to less than 500, while still keeping relative distances. This reduction pulls together related documents and terms, while removing the background noise. Ding [10] observes that there is no rule for selecting the number of dimensions to represent in Latent Semantic space, but a number between 300 and 1000 is usually suitable.

### 3.1 Singular Value Decomposition

Singular Value Decomposition is a matrix factorisation that is at the heart of Latent Semantic Analysis. It removes the noise from the matrix, and draws together related concepts within the matrix.

The process involved in Singular Value Decomposition is one of creating a term-collection matrix is created, adding the query to the matrix in the form of a new (small) document column. As the matrix will be sparse (typically less than 1% full) a form of Harwell-Boeing Sparse Matrix will be used to store the data. The query is represented as a small collection in the matrix. Terms within the query can also be given weights as to how important they are to the query, allowing relevance feedback to be implemented. Negative weights can be given to terms that are not to be returned in the query. Applying Singular Value Decomposition to the matrix returns a term matrix (U), a Singular Value matrix (S), and a collection matrix (V). The term matrix shows the high level statistical relationships between the terms in the collection, while the collection matrix shows the high level statistical relationships between each of the collections.

At this point a term-term or collection-collection matrix can be created, each showing the relationship with its members. A search can be performed on the collection-collection matrix to find the highest cosine related collection. This also means that a collections can be returned that did not include the query terms in the collection. This is one of the most important results of Single Value Decomposition, and it defeats the problems of both synonymy and polysemy. [1].

Deerwester [8] give the following formula which shows

how Singular Value Decomposition is calculated.

$$A = U \Sigma V^T$$

where,

- A is our  $m \times n$  matrix
- U is a  $m \times m$  orthogonal matrix
- V is a  $n \times n$  orthogonal matrix
- $\Sigma$  is an  $m \times n$  diagonal matrix where element  $s_{ij} = 0$  if  $i <> j$ , and  $s_{ii} \geq 0$
- The quantities  $s_{ii}$  are the singular values of A
- $V^T$  indicates the transpose of V

As an example, the following matrix was decomposed into its component parts using the SVD function in MATLAB 6.1.

$$A = \begin{bmatrix} 0.96 & 1.72 \\ 2.28 & 0.96 \end{bmatrix}$$

Singular Value Decomposition of A is:

$$A = U \Sigma V^T = \begin{bmatrix} 0.6 & -0.8 \\ 0.8 & 0.6 \end{bmatrix} \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0.8 & 0.6 \\ 0.6 & -0.8 \end{bmatrix}$$

These matrices can now be used to compare collections and terms together.

### 3.2 Collection Selection Algorithm

In this section, we give the details of our collection selection algorithm. The inputs of the algorithms include a query, a selected set of terms (key words), and a set of sample documents from each collection.

#### Algorithm 3.1

1. Calculate the term-collection matrix A where we view the query as a new collection
2. Use singular value decomposition  $U \Sigma V^T = A$
3. Sort the collections according to the values in the query row in the matrix  $V^T$
4. Use the threshold to calculate a rank of collections

The important and difficult problem here is to calculate the term-collection matrix  $A$  because we cannot directly use the  $tf \times idf$  technique. We have to consider term distribution in the collections because it is natural that terms distributed across all documents in a collection are worth more than terms clustered in one document.

In the following we will show our approach for calculating the term-collection matrix  $A$

Let  $C_j = \{d_{j1}, \dots, d_{jn_j}\}$  ( $1 \leq j \leq n$ ) be a collection, and  $K = \{t_1, \dots, t_m\}$  be a selected set of key words. Then for a given term  $t_i$  and a collection  $C_j$  we can define the importance of term  $t_i$  in the collection  $C_j$  as follows:

$$a_{ij} = \sum_{k=1}^{n_j} \begin{cases} 1 + \frac{f_{i,d_{jk}}}{\Delta}, & \text{if } t_i \text{ appears in document } d_{jk} \\ 0, & \text{otherwise} \end{cases}$$

where  $f_{i,d_{jk}}$  is the frequency of the term  $t_i$  in document  $d_{jk}$ , and

$$\Delta = \sum_{s=1}^m \sum_{t=1}^n \sum_{u=1}^{n_t} f_{s,d_{tu}}$$

## 4 Term/Collection Co-Relations

We have developed a prototype system using Latent Semantic Analysis. This system provides promising results.

The user enters a query into the prototype interface, which is then sent to a diverse set of search engines in parallel, and the top  $n$  results from each engine ( $n$  is currently 10) are appended to a file.

On the first pass of the collection file (a set of appended documents) the parser extracts the keywords, removes all stopwords (common words such as 'a', 'the', 'their') and then sorts the keywords.

On the second pass of the collection file, the parser builds an inverted index from a set of documents. An inverted index is a file that contains a list of all terms, the inverse document frequency, their frequencies, and normalised frequencies, and term weighting.

The term weighting schema chosen can make a large difference on the quality of the matrix operations. Term weighting is a process of removing bias to large documents, and small and large collections. It effectively reduces the term to a representative value of the documents. The researcher is evaluating the modified Okapi [2] weighting scheme in this system, which for collections is calculated as:

$$\frac{tf}{0.5 + 1.5 \frac{cl}{avgcl} + cf} \log \frac{N - cf + 0.5}{cf + 0.5}$$

where  $tf$  is the term frequency,  $cf$  is the collection frequency,  $cl$  is the collection length (total size of all documents

in a collection),  $N$  is the total number of documents in the collection, and  $avgcl$  is the average collection length.

The query is now added to the matrix. The query is considered to be a very small collection. If query expansion is being used a small representative document may be added to the matrix instead of a query.

A term-collection matrix is then generated. Singular Value Decomposition, the primary function of Latent Semantic Analysis, is applied to this matrix, followed by matrix column normalisation and transformation. The resulting matrix provides a relationship between each of the collections and the query. The most related collections will be closest to the number 1. The least related collections will be closest to the number 0.

The results from this prototype are promising, however as this is a new method some of the results are unusual and difficult to explain, such as some of the relationships produced during decomposition.

## 5 Precision/Collection Co-Relations

A collection selection system needs to quickly learn which collections contain information relevant to the user and which collections contain information relevant to the user. A high quality subset of collections can be used to help improve precision and to cut down on bandwidth and computational costs. The solution presented here will make use of user ranked relations between collections.

Collection relations can be used to reduce the number of collections needing to be searched over time. The reasoning behind this is that information in documents tend to cluster together, which is widely utilised by search engines that rely on hubs and authorities as central sources of quality information [14]. This research proposes that the same clustering that occurs among documents tends to occur in collections, especially in small to medium sized collections, thus finding relationships between collections is a viable method of returning better results.

For every search performed, the user will give the top  $n$  collections ( $n$  is currently 10) a floating point precision ranking in the range of 0 to 1. The higher the ranking the more precise the results. After a training run of (say) twenty searches singular value decomposition will be applied to the precision/collection matrix and the latent statistical relationships between collections computed. The returned values are a score for each collection, with zero being not relevant and one being most relevant. This will find relationships existing between collections that are not immediately obvious, and will result in a more personalised search which will over time learn the users preferences.

After the training is completed, every time the user enters a query the collection/term matrix will be queried to find the highest rated collection to search. All collections

Collection	CiteSeer	ProQuest	ACM	AusStats
Docs Used	10	15	10	10
User Feedback	0	1	2	0
Term/Coll	0.56	0.87	0.97	0.92
Precision/Coll	0.15	0.95	0.74	0.67

TABLE 1: *The results of a sample search of four collections. It can be seen that Term/Collection can be different to the user rated Precision/Collection results. This is an example of the system learning the users preferences*

closely related to this collection will be looked up in the precision/collection matrix and the top  $n$  matrices will be queried. This will mean that the system will quickly learn related collections, and will only have to return the most relevant collections for searches, instead of having to query all collections. This will reduce number of collections sampled. Further collection rankings will be folded into the matrix, and the matrix re-computed every few searches from then on. The value of this information will increase over time, as more relationships between collections are learned. After a large number of iterations earlier searches could be dumped from the matrix to prevent skew to collections no longer producing relevant results.

Early tests in MATLAB indicate that this is an excellent method for finding correlations among collections, and for selecting the most relevant collections. This promises to allow feedback from the user to improve collection selection.

## 6 Testbed

The experiments are conducted using ten popular collections. Each of these engines returned ranked results to a query. The results are shown in Tables 1, 2, and 3.

The collections used are CiteSeer, ProQuest, ACM, AusStats, ScienceDirect, Web of Science, SwetsWise, Ebsco, Informit Online, and Altavista.

The experimental methodology being emulated is that of Callan *et al* [3], who tested five database ranking algorithms on a large testbed of data under different conditions. Conditions to be changed are resource descriptions (complete and sampled), query size (long and short), and sample size (5, 10, 20, 30, 40, 50, 100, 150 documents). The desired collection rankings will be compared against the actual collection rankings using the collection precision and recall metrics. The measure of success will be that the desired collection rankings match the collection rankings. A second test is made using the merged result using the document precision and document recall metrics. A high recall and precision will indicate a effective collection selection algorithm.

	G	AV	SD	ACM	CS
Google(G)	1.00				
Altavista(AV)	0.28	1.00			
SD	<b>0.88</b>	0.25	1.00		
ACM	0.353	0.35	0.34	1.00	
CiteSeer(CS)	0.31	0.30	0.31	<b>0.39</b>	1.00

TABLE 2: *The sample relationships between five of the collections for the "computer science" collection. From this table it is easy to see the highest related collections. For example, Google is highly related to Science Direct(SD), with a score of 0.88.*

Collection	SVD Ranking	User Ranking
Google	0.78	0.73
ScienceDirect	0.57	0.60
ACM	0.40	0.47
CiteSeer	0.35	0.60
Altavista	0.27	0.57

TABLE 3: *The sample results of a SVD and User ranking of five collections across several different queries. The SVD and User results both show that Google is the best collection.*

## 7 Conclusions

A solution to the Web Based Collection Selection problem has been presented, and preliminary results indicate that the technique is suited to the task of selecting the most relevant collections and learning user preferences in collections. The approach uses short queries and is thus suitable for use on the Web. This approach also reduces the need for ontologies and thesaurus.

With some modification, this collection selection method is suitable for traditional information retrieval systems across servers and databases. A problem is that these systems do not rank the data before returning it. This could be solved using simple sampling techniques that would grab a representative sample of the collection, rank it, than compare it across collections.

As the number of collections indexed grows, so does the number of terms and the size of the matrix. The Achilles heel of SVD is that as the matrix size increases, so does the time it takes to process the matrix, and it is not suitable for large numbers of documents. However in this research, only the top  $n$  most representative documents from each collection are sampled so it is possible to compare hundreds of collections in a reasonable time if  $n$  is small.

Due to the time expense of writing screen scraping applications for web based collections and comparing the results to human rankings of the documents in the collections, the researchers were unable to perform large scale tests of the

methods presented in this research.

Work still needs to be done to find the optimal term weighting scheme, and on the optimal sample size taken from each collection.

## References

- [1] M. W. Berry, Z. Drmac, and E. R. Jessup. Matrices, vector spaces, and information retrieval. volume 41, pages 335–362. Society for Industrial and Applied Mathematics, 1999.
- [2] C. Buckley, A. Singhal, M. Mitra, and G. Salton. New retrieval approaches using smart: Trec. In *In Proceedings of the Fourth Text Retrieval Conference (TREC-4)*, pages 25–48. NIST Special Publication 500- 236, October 1996.
- [3] J. Callan, A. L. Powell, J. C. French, and M. Connell. The effects of query-based sampling on automatic database selection algorithms. Technical Report Technical Report CMU-LTI-00-162, Language Technologies Institute, School of Computer Science, Carnegie Mellon University, 2000.
- [4] J. P. Callan, Z. Lu, and W. B. Croft. Searching Distributed Collections with Inference Networks . In E. A. Fox, P. Ingwersen, and R. Fidel, editors, *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–28, Seattle, Washington, 1995. ACM Press.
- [5] Y. S. Choi and S. I. Yoo. Neural net agent for discovering text databases on the web. In J. Eder, I. Rozman, and T. Welzer, editors, *Advances in Databases and Information Systems, Third East European Conference, ADBIS'99, Maribor, Slovenia, September 13-16, 1999, Proceedings of Short Papers*, pages 221–231. Institute of Informatics, Faculty of Electrical Engineering and Computer Science, Smetanova 17, IS-2000 Maribor, Slovenia, 1999.
- [6] N. S. Chung-Min Chen. Telcordia lsi engine: Implementation and scalability issues. page 51, Heidelberg, Germany, April 01 - 02 2001. 11th International Workshop on research Issues in Data Engineering, Heidelberg, Germany.
- [7] N. Craswell, P. Bailey, and D. Hawking. Server selection on the world wide web. In *Proceedings of the fifth ACM conference on Digital libraries, San Antonio, Texas, United States*, pages 37–46. ACM Press, 2000.
- [8] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.
- [9] C. H. Q. Ding. A similarity-based probability model for latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, Berkeley, California, United States*, pages 58–65. ACM Press, 1999.
- [10] C. H. Q. Ding. A similarity-based probability model for latent semantic indexing. In *Research and Development in Information Retrieval*, pages 58–65, 1999.
- [11] J. C. French, A. L. Powell, J. P. Callan, C. L. Viles, T. Emmitt, K. J. Prey, and Y. Mou. Comparing the performance of database selection algorithms. In *Research and Development in Information Retrieval*, pages 238–245, 1999.
- [12] L. Gravano and H. García-Molina. Generalizing GLOSS to vector-space databases and broker hierarchies. In *International Conference on Very Large Databases, VLDB*, pages 78–89, 1995.
- [13] L. Gravano, H. García-Molina, and A. Tomasic. GLOSS: text-source discovery over the Internet. *ACM Transactions on Database Systems*, 24(2):229–264, 1999.
- [14] J. M. Kleinberg. Hubs, authorities, and communities. *ACM Computing Surveys (CSUR)*, 31(4es):5, 1999.
- [15] Z. Lu, J. Callan, and W. Croft. Applying inference networks to multiple collection searching. Technical Report TR96–42, University of Massachusetts at Amherst. Department of Computer Science, 1996.
- [16] Z. Lu and K. S. McKinley. *Advances in Information Retrieval*, chapter The Effect of Collection Organization and Query Locality on Information Retrieval System Performance and Design. Kluwer, New York, New York, 2000.
- [17] F. Menczer and R. K. Belew. Adaptive retrieval agents: Internalizing local context and scaling up to the web. *Machine Learning*, 39(2/3):203–242, 2000.
- [18] E. Voorhees, N. Gupta, and B. Johnson-Laird. The collection fusion problem. In *In Proceedings of TREC-3*, pages 95–104, 1995.