Web Search Result Clustering using Heuristic Search and Latent Semantic Indexing

Mansaf Alam
Dept of Computer Science
Jamia Millia Islamia
New Delhi, India

Kishwar Sadaf
Dept of Computer Science
Jamia Millia Islamia
New Delhi, India

ABSTRACT
Giving user a simple and uncomplicated web search result representation is an active area of Information Retrieval research. Traditional search engines use the hyperlink structure of the web to retrieve documents or pages and give them in a ranked fashion to the user. In this paper, we propose a technique for grouping web search results into meaningful clusters. The proposed method performs heuristic search on the query result graph to prune undesired edges to form cluster and carries out Latent Semantic Indexing within these clusters to make them refined, meaningful, and relevant to the query.

General Terms
Document Clustering, Heuristic Search, Semantic Similarity, LSI et al.

Keywords
Web search, clustering, heuristic search, LSI, web graph.

1. INTRODUCTION
The information available on the web is unstructured, disorganized, dynamic and heterogeneous in nature. Moreover the process of retrieval is highly affected by the ill formed query put up by the average user. Today's search engines return too many results which are not necessarily relevant to the user's need. Usually a user has to traverse several search result pages to get to the desired result. A way of assisting users in finding what they are looking for quickly is to group the search results by topic. The user does not have to reformulate the query, but can merely click on the topic most accurately describing his or her specific information need. This grouping of result is called Clustering. More specifically, it is a process of grouping similar documents into clusters so that documents of one cluster are different from the documents of other clusters. Web search result clustering has been the focus of IR community since the emergence of web search engine. Therefore numerous works has been done in this area. The Scatter/Gather system by Cutting et al [1] is one of the pioneer works and conceptual father of all web search result clustering systems. It's based on two clustering methods namely Buckshot and Fractionation. There are many web clustering engines available on the web (Carrot2, Vivisimo, SnakeT, Grouper etc) which give the search results in forms of clusters. The main drawback of many web clustering engines is that they take into account only the topical similarity between documents that are returned by the search engine. A common technique used by search result clustering engines is to cluster so-called document snippets rather than entire documents. Snippets are the small paragraphs often displayed along with web search results to give the user a hint of the document contents. A web search result clustering engine takes the result i.e., returned by the search engine as input and performs clustering and labeling on that result. They are usually seen as complementary rather than alternative to the search engine [2]. The main use for web search result clustering is not to improve the actual ranking, but to give the user a quick overview of the results. Users want whole picture of their search result. Having divided the result set into clusters, the user can quickly narrow down his search further by selecting a cluster. An important text-based clustering, Suffix Tree Clustering (STC), is based on the Suffix Tree Document (STD) model which was proposed by Zamir et al [3]. The similarity between documents is based on matching phrases rather than on single words only. A phrase in this context is an ordered sequence of one or more words. The STC algorithm focuses on clustering document snippets returned by the search engine, faster than standard data mining approaches. They build Grouper – a clustering interface to the HuskySearch meta-search engine, which uses STC as its clustering algorithm. There are numerous works which have been derived from STC algorithm [4, 5, & 6]. Yao et al [7] present a token-based web snippet clustering. A snippet is composed of several tokens. These token are used as basic units for clustering. Sha et al [8] propose a web search result clustering based on lexical graph. Authors show that lexical graph structure is suitable in finding the word relationship and synonyms. The search result is partitioned according to the graph where the large degree node becomes the center of the cluster. Orthogonal clustering can also be applied in web search clustering [6]. The authors propose a Semantic, Hierarchical, Online Clustering (SHOC) approach to cluster web search result. The approach consists of three phases: Data collection and cleaning, Feature extraction & Identifying and organizing clusters. Kummamuru et al [9] notice that monothetic document clustering algorithm is best suited for web search result summarization and browsing of the search result. The proposed technique works by defining an optimality criterion and progressively selecting new concepts from a set of candidates at each level of the hierarchy. The selection process attempts to maximize coverage (defined as the number of documents covered by a concept) and maintain distinctiveness between the candidate concept and other concepts on the same branch of the hierarchy.

Zeng et al [10] propose a web search clustering technique wherein phrases from snippets are extracted and ranked to form cluster labels. The documents are assigned to relevant clusters labels to form initial candidate clusters and then these candidate clusters are merged to form final clusters.

The paper is organized as follows. In section 2, related works are introduced. Some important methods are identified in section 3. In section 4, proposed approach is explained. Finally we derive the conclusion in section 5. Page, node and document are used interchangeably throughout the paper.

2. RELATED WORK

The web search result clustering has been investigated in a number of earlier works starting from Scatter/Gather system. Some of them are [11, 12]. The problem of clustering web search result using web graph and heuristic search is quite new to the area of IR. As far as we know, Bekkerman et al [13**]** is the only work based on heuristic search in web graph. Our proposed method makes use of this work. Mecca et al [14] use Singular Value Decomposition (SVD) on documents returned by the search engine as a whole instead of document snippets. Our approach differs from this work. We first cluster the documents and then apply LSI (which uses SVD) to each cluster to make it more coherent and refined.

3. BACKGROUND

Giving user a simple and uncomplicated web search result representation is an active area of Information Retrieval research. Search engine like Google uses the hyperlink structure of the web to retrieve documents or pages and give them in a ranked fashion to the user. This hyperlink structure is basically a graph, where a node represents a page and a link is characterized by an edge.

3.1 Link Analysis

Link analysis uses information about the structure of the web graph to aid search. Using links, collection of documents or pages can be viewed as a graph. The structure of this graph, independent of content, can provide interesting information about the similarity of documents and the structure of the web. The pioneer works in the field of link-based web search are [15] and [16]. They have inspired many other works. [16] is based on hyperlink structure of the web pages. Authors of Hits propose that there are two kinds of pages in search results: Hub and Authority. Authorities are pages that are recognized as providing valuable, significant, reliable, and useful information on a topic. Hubs are index pages that provide lots of useful links to relevant content pages (topic Authorities) i.e. their role is advertise authority pages. However, if the document collection consists of several topics, authority and hub pages may only cover the most popular subjects and leave out the less popular ones. PageRank uses an alternative link-analysis method. It ranks pages using authority. Its applied to the entire web rather than a local neighborhood of pages surrounding the results of a query. Many commercial search engines such as Google (based on PageRank) and Yahoo are being used by people across the globe, the relevancy of documents returned in the search engine result still lacks. That's why rigorous researches are being carried out in the field of IR. The objective of a search engine must be to satisfy the user's information need in a lucid way. Clustering techniques are undeniably useful in this context. They have the potential to group similar documents into cohesive factions.

Applying clustering on the hyperlink structure of web documents is a promising area in IR research. Clustering has become a vital technique for web search results grouping into meaningful and significant clusters. There are many clustering engines available, like Kartoo, Carrot, SnakeT etc. Further research in this area is being done to make information retrieval more relevant. Leuski et al [17] propose a method where ranking and clustering are combined. The approach first traverses through the ranked list returned by the search engine until a relevant document is found. This document is then used as a cluster seed and clustering is performed on unexamined documents. The result set returned by the search engine preserves the essential properties of the entire hyperlink structure used by search engine. Wang et al [18] propose a web search result clustering which makes use of the hyperlinks between the pages and employs the HITS algorithm and k-means clustering. Many studies [19 & 20] show that the web graph (the underlying hyperlinked graphical structure of web documents) follows the power-law distribution of degrees. This phenomenon has become a basic web graph property. In [21], authors analyze a subset of the web graph- web of Spain and observe that this subgraph follows the power-law distribution same as the web graph. Based on this web property, Bradic [22] shows that the underlying graph of the search result is basically a subgraph of the whole web graph and follows the same degree distribution. He applies graph partitioning method using random walk on this subgraph to form clusters.

3.2 Heuristic Search

Heuristic Search is an Artificial Intelligence searching technique that applies heuristics. Heuristic is a rule of thumb. It exploits additional knowledge about the problem that helps direct search to more promising paths. Heuristics help to reduce the number of alternatives from an exponential number to a polynomial number.

Heuristics can play a major role in searching optimal paths in the web search result graph exponential nature of the web. Bekkerman et al [13] propose a multiagent, and bidirectional based heuristic search in the web graph. They apply beam search in the search result graph in parallel to traditional topical clustering method on the clusters so formed. Hybrid approach like [18] and [13] has not been used in its full potential. It may prove costly in term of time but the resultant clusters will be topologically and topically similar.

All the methods mentioned above only consider lexical similarity between documents. That means if a document doesn't have lexical words same as documents in a cluster yet having same meaning is not welcome to the cluster. The method Latent Semantic Indexing (LSI) overcomes this problem. It groups words that are semantically same. Our proposed method uses this technique.

3.3 Latent Semantic Indexing

The Latent Semantic Indexing (LSI) is the machine-learning technique that identifies, represents, and compares concepts existing within a collection of documents or data. Latent Semantic Indexing [23], also known as Latent Semantic Analyses (LSA) means analyzing documents to find the underlying meaning of those documents. LSI is a pure mathematical method. Although the LSI algorithm doesn't understand anything about what the words mean, the patterns it detects can make it seem surprisingly intelligent.

LSI arose from the problem of how to find relevant documents out of million ones from user query words. The fundamental difficulty arises when we compare words to find relevant documents, because what we really want to do is compare the meanings or concepts behind the words. English language in general, is very amusing mostly because of synonymous and polysemous words. Search engines cannot assume the exact meaning of the synonymous or polysemous query words. LSI attempts to solve this problem by mapping both words and documents into a "concept" space, with "latent" semantic dimensions and doing the comparison in this space. The latent semantic space that we project query and documents into has fewer dimensions than the original space (which has as many dimensions as terms). LSI is thus a

method for dimensionality reduction. A dimensionality reduction technique takes a set of objects that exist in a high-dimensional space and represents them in a low-dimensional space, often in a two-dimensional or three-dimensional space for the purpose of visualization. Latent semantic indexing applies a particular mathematical technique, called Singular Value Decomposition or SVD, to a word-by-document matrix. SVD project an n-dimensional space onto a k-dimensional space where n ≫k. In our application (term-document matrices), n is the number of word types in the result set. The projection transforms a document's vector in n-dimensional word space into a vector in the k-dimensional reduced space.

The SVD is computed by decomposing the word-by-document matrix $A_{t\times d}$ into product of three matrices, $T_{t\times d}$, $S_{n\times n}$ and $D_{d\times n}$:

$$A_{t\times d} = T_{t\times d}\ S_{n\times n}\ (D_{d\times n})^T$$

where t is the number of terms or words in the document set, d is the number of documents, $n = min(t, d)$, T and D have orthonormal columns, i.e. $TT^T = D^T D = I$, and $rank(A) = r$, singular values on the diagonal of the matrix $S = diagonal(\sigma_1, \sigma_2, \cdots \sigma_n)$. $\sigma_i = 0$ for $1 \le i \le r$, and $\sigma_j = 0$ for $j \ge r + 1$ [24].

SVD can be viewed as a method for revolving the axes of the n-dimensional space such a way that the first axis runs along the direction of largest variation among the documents; the second dimension runs along the direction with the second largest variation and so forth. The T matrix gives us the coordinates of each word on our "concept" space, the D^T matrix gives us the coordinates of each document in our "concept" space, and the S matrix of singular values gives us a clue as to how many dimensions or "concepts" we need to include. By truncating the matrices T, D^T and S to their first k< n rows, we get the matrices $\boldsymbol{T_{t\times k}}$, $\boldsymbol{S_{k\times k}}$ and$(\boldsymbol{D_{d\times k}})^{\boldsymbol{T}}$. Their product $\boldsymbol{A_{t\times k}}$

$$A_{t\times k} = T_{t\times k}\ S_{k\times k}\ (D_{d\times k})^T$$

is the best projection of original matrix A into a k-dimensional space. Choosing the number of dimensions (k) for A is an interesting problem. A small k can remove much of the noise, but too few dimensions may lose important information. The parameter k should be large enough to allow fitting the characteristics of the data and small enough to filter out the non-relevant representational details. LSI works well with a relatively small number of dimensions (k) [25]. The query is also represented as a vector in a *k*-dimensional space

$$\hat{q} = q^T\ T_{k\times k}\ {S^{-1}}_{k\times k}$$

where q is the vector of words in the users query, multiplied by the appropriate term weights. The sum of these k-dimensional terms vectors is reflected by the term $\boldsymbol{q^T T_{k\times k}}$ in the above equation, and the right multiplication by $\boldsymbol{S^{-1}_{k\times k}}$ differentially weights the separate dimensions. Thus, the query vector is located at the weighted sum of its constituent term vectors. This query vector is then compared to all documents in the dataset to determine the proximity between query and documents.

4. PROPOSED METHOD

In this paper, we propose a method for clustering web search result using heuristic search and LSI. This is the first application of LSI for clustering the web search result involving heuristic search as far as we know. Our goal is to present a cluster in a list form where tightly semantically related pages are near to each other. We are not concerned with the effectiveness of the search engine, but the way the result is presented. Search engine like Google takes many factors into consideration other than hyperlink like determining the popularity of the page, the position and size of the search terms within the page, and the proximity of the search terms to one another on the page. But still many a times a user has to traverse through several search engine pages to get to the relevant page, which is quite cumbersome and a novice user seldom traverse to fourth or fifth page of the search result. We need a system which identifies the user information need and give the same on the first page of result. Our method tries to do that by combining heuristic search and LSI. Much about LSI and heuristic search is given above.

First, we use heuristic search to form clusters. Bekkerman et al [13] is the only work so far, as we know, that employs heuristic search on the web graph. We use this idea of heuristic search only in pruning the undesired links of web graph that is preserved in the search result to form clusters. Initially each node of the result set web graph is assigned an agent and each node is assigned to a singleton cluster. Each agent maintains a list which refers to all its connected nodes. Each agent selects a seeker node from the list which is most promising. Initially the source node is the seeker node. At each iteration, an agent selects a seeker node to find its connected nodes and add them to the list. This selection of seeker node is made using heuristic: find a node with high number of links. If a node has many links, implies that there are many edges between nodes. Clusters in a graph can be identified by high number of edges within and less number of edges between them. Unlike Bekkerman's heuristic, we select a node to be seeker if it would lead to more connected nodes. After that the lists obtained by agents are intersected. If a common node is found in two lists, we merge the clusters that their source nodes belong to. In this way we get pages that are topologically similar. To get coherent, refined and meaningful clusters with features more related to the query, we use LSI which transforms all the pages or documents in a cluster, into LSI feature vectors. The query is also represented as a vector in a *k*-dimensional space. The query vector can then be compared to all existing document feature vectors in a cluster, and the documents are ranked by their similarity (nearness) to the query. One common measure of similarity is the cosine between the query vector and document vector. We would use this method. This comparison of query with all the document vectors in all clusters will be done parallel. The document which has the highest similarity with the query is positioned first in the cluster list, since query is the only clue to the user information need. Other documents are placed according to their similarity with the query, in the list, in decreasing order. The cluster with the highest number of members is offered on the first page of the Clustering system.

We describe our method step-by-step as follows:

Step 1: Parse the search result and form a hyperlink graph.

Step 2: Assign an agent to each node of the graph.

Step 3: Each agent maintains a list of connected nodes. Initially the list contains the source node or page. Each agent

selects a promising node "Seeker node", which, at each iteration is to be searched for more connected nodes. Initially the source node is the seeker node. Also each node is assigned to a singleton cluster.

Step 4: At each Iteration, an agent renews the list by selecting a seeker node using heuristic: find a node with high number of links. If a node has many links, implies that there are many edges between nodes. Clusters in a graph can be identified by high number of edges within and less number of edges between them.

Step 5: After that, all the lists obtained by each agent are intersected. If a common node found for two source nodes, the clusters they belong to, are merged. This step continues for predefined number of iterations. At the end of this step, we get clusters that are topologically similar.

Step 6: Extract the contents of nodes of each cluster and **preprocess** them. Preprocessing involves stop-word removal, tokenization, stemming etc.

Step 7: For every document set in a cluster, build a term-document matrix.

Step 8: Compute SVD for each cluster.

Step 9: Truncate SVD into reduced-K LSI space.

Step 10: Map the query to reduced K-space.

Step 11: Determine documents proximity with query using cosine method for each cluster.

Step 12: The document with the highest similarity with the query is positioned first in the cluster list. The cluster with the highest number of members is offered as expanded on the first page of the Clustering system.

On the basis of above defined steps, we describe our proposed method algorithmically as follows:

Search Result Acquisition

Form graph of the search result using hyperlink.

Let $P = \{p_1, p_2 \cdots, p_n\}$, nodes of the graph

Assignment of agents to each node of the graph

For each node $p_i \in P$

Assign agent $a_i \; to \; p_i : a_i \, (p_i)$

Initialize agenta_i's seeker_node $S(a_i) \leftarrow p_i$

Initialize agenta_i's list_of_nodes $L_j \, (a_i)$

Search Phase

For each node $p_i \in P$

Expand list_of_node L

$L_j \, (a_i) \leftarrow$ connected node $S_j(a_i)$

Filter $L_j \, (a_i)$ for new seeker using heuristics

Update seeker $S_j \, (a_i)$

Cluster Construction phase

Initialize singleton Cluster $C_i \leftarrow p_i$

Construct all pairs$(p_i, p_{i'}) \;\; s.t. \;\; L_j \, (a_i) \cap \; L_j \, (a_{i'}) = \emptyset$

For each node pair $(p_i, p_{i'})$

If $(p_i \in C_m) \cap (p_{i'} \in C_{m'}) \cap (C_m \neq C_{m'})$ then

Merge C_m and $C_{m'}$

Cluster Refinement

Let $k =$ LSI space

Reform Query q as $\hat{q} = q^T \, T_{k \times k} \, {S^{-1}}_{k \times k}$

For each cluster $c_i \;\; \in \;\; C$ do

For each node $p_j \; \in \; c_i$ do

Extract content of each node

$d_j \leftarrow p_j$

Preprocess d_j and extract terms into T

Construct term-document matrix $M_{i(|T| \times |c_i|)}$

Decompose M_i using SVD

Perform **LSI** by truncating **SVD** into k-LSI space

Map $\hat{q}$ into k-LSI space

Measure similarity between query vector and document

$d_j \; cosine \; \hat{q}$

Positioning of Result

For each document d_j of the cluster

Place document into the cluster list according to its similarity with query q

Our main goal is to combine the powers of links and clustering to give user relevant result in an effective format. Our approach takes care of the topological as well as semantic property of the documents. Semantic Similarity relates to computing the similarity between concepts which are not necessarily lexically similar. Pages with no to low "lexical" similarity with query end up last in the ranked search result of the search engine but that doesn't make them less relevant to the query. They may be conceptually similar to the query. LSI is a method which retrieves documents based on concept matching rather than index term matching. Matching based on concept allows documents to be retrieved even though they are not indexed by the query index terms. The advantage of our approach is its usefulness as it takes into account the whole document rather than just small snippets. A two or three lines of snippet cannot provide the whole perspective of the document. One of the drawbacks of ranked result system

is the dependence of result on the lexical similarity between user query and documents. Documents with significant semantic similarity but less term-to-term similarity with query fall deeper in the search result where users hardly take pains to dig. Other web search clustering methods that exploit document snippets that are returned by the search engines, have only limited feature of documents and only lexical similar terms of query. A document which has insignificant lexical similarity but semantically similar with query fails to become a member of a relevant cluster. Our method clearly overcomes this limitation. Our method not only utilizes the strong link structure of the web documents to create clusters but also applies powerful technique LSI to claim semantic similarity The computing overhead of using whole document is diminished by the fastness of LSI.

5. CONCLUSION AND FUTURE WORK

A user wants to quickly locate his/her information need and search engine result list seems endless to the user. Although similar documents tend to link each other, the ranked result doesn't give a compact and precise result. Clustering can solve this problem. It's is an empowering tool that is yet to be used in its full potential. Heuristic Search is well suited for the domain of web documents since there are multiple links or paths between documents and it can be used in pruning of undesired links. It is advantageous to cluster documents in a reduced dimension LSI space rather than term space, because original term space is not a good representation of the document collection as a whole. Two documents may be semantically very close even if they do not share a particular keyword, LSI does not require an exact match to return useful results. Where a plain keyword search in term space will fail if there is no exact match, LSI will often return relevant documents that don't contain the keyword at all. For future work, we are planning to implement this new approach on a real system. It would be interesting to see the actual results. Also we are planning to give the clusters meaningful labels so that users can get to a particular cluster directly. Labeling is of immense importance since a good and up to point label gives the exact account of what a cluster holds.

6. REFERENCES

[1] Cutting, D.R., Kager, D.R, Pedersen, J. O., and Tukey, J.W. 1992. Scatter/gather: a cluster-based approach to browsing large document collections. The 15th annual international ACM Sigir conference on Research and development in information retrieval, pp.318-329.

[2] Carpenito, C., Osinski, S., Romano, G. and Weiss, D. 2009. A Survey of Web Clustering Engines. ACM Computing Surveys, Vol. 41, No. 3, Article 17.

[3] Zamir O. and Etzioni, O. 1998. Web document clustering: A feasibility demonstration. In Research and Development in Information Retrieal ,1998, pp. 46-54.

[4] Branson, S. and Greenberg, A. 2009. Clustering Web Search Results Using Suffix Tree Methods. Stanford university.

[5] Janruang, J. and Guha, S. 2011. Semantic Suffix Tree Clustering. First IRAST International Conference on Data Engineering and Internet Technology (DEIT).

[6] Zhan, D. and Dong, Y. 2004. Semantic, Hierarchical, Online Clustering of Web Search Results. Advanced Web Technologies and Applications 6th Asia-Pacific Web Conference, APWeb 2004, Hangzhou, China.

[7] Yao, T. and Li, J. 2006. A Token-based Online Web-Snippet Clustering Approach based on Directed Probability Graph. Second International Conference on Semantics, Knowledge and Grid.

[8] Sha, Y. and Zhang, G. 2009. Web search result clustering algorithm based on lexical graph. Journal Of Computational Information Systems,Volume: 5, Pages: 283-290.

[9] Kummamuru, K., Lotlikar, R., Roy, S., Singal, K. and Krishnapuram, R. 2004. A Hierarchical Monothetic Document Clustering Algorithm for Summarization and Browsing Search Results. In Proceedings of the 13th international conference on World Wide Web.

[10] Zeng, H., He, Q., Chen, Z., Ma, W. and Ma, J. 2004. Learning to Cluster Web Search Results. In Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information.

[11] Hearst, M. A. and Pedersen, J.O. 1996. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. SIGIR-96th ACM International Conference on Research and Development in Information Retrieval pp. 76-84.

[12] Leouski, A. and Croft, W.B. 1996. An evaluation of techniques for clustering search results. Technical Report IR-76, University of Massachusetts, Amherst.

[13] Bekkerman, R., Zilberstein, S. and Allan, J. 2007. Web Page Clustering using Heuristic Search in the Web Graph. Proceedings of IJCAI-07, the 20th International Joint Conference on Artificial Intelligence.

[14] Mecca, G., Raunich, S. and Pappalardo, A. 2007. A New Algorithm for Clustering Search Result‖. Journal of Data & Knowledge Engineering Volume 62 Issue 3, September.

[15] Brin, S. and Page, L. 1998. The anatomy of a large-scale hypertextual web search engine. In Proceedings of WWW7, Brisbane, Australia.

[16] Kleinberg, J.M. 1998. Authoritative sources in a hyperlinked environment. In proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms (SODA).

[17] Leuski, A. and Allan, J. 2000. Improving Interactive Retrieval by Combining Ranked Lists and Clustering. In Proceeding of RIAO, pp.665-681.

[18] Wang, Y. and Kitsuregawa, M. 2001. Link Based Clustering of Web Search Results. In Proceedings of The Second International Conference on Web-Age Information Management (WAIM2001), Xi'An, P.R.China, Springer-Verlag LNCS..

[19] Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A. and Wiener, J.2000. Graph structure in the web : Experiments and model. Proceedings of the Ninth Conference on World Wide Web, pp 309-320.

[20] Barabasi, A. and Albert, R. 1999. Emergence of Scaling in Random Networks. Science 286 (509).

[21] Baeza-Yates, R., Castillo, C. and Lopez, V. 2005 Characteristics of the Web of Spain. Cybermetrics, Vol. 9, No. 1.

[22] Bradic, A. 2009. Search Result Clustering via Randomized Partitioning of Query-Induced Subgraphs. Telfor Journal, Vol.1, No.1.

[23] Deerwester, S., Dumais, S.T., Furnas, G., Landauer, T. and Harshman, R. 1990. Indexing by Latent Semantic Analysis. Journal of the American Society for Information Science, pp. 391-407.

[24] Rosario, B. 2000. Latent Semantic Indexing: An overview. In Proceedings of Infosys, vol. 4.

[25] Berry, M.W., Dumais, S.T. and O' Brien, G.W. 1995. Using linear algebra for Intelligent Information Retrieval. SIAM.