Open Access Library Journal
2023, Volume 10, e10228
ISSN Online: 2333-9721
ISSN Print: 2333-9705

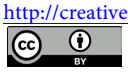
Enhanced Document Retrieval System Using Suffix Tree Clustering Algorithm

Linda Uchenna Oghenekaro, Ifeanyi Emmanuel Olughu, Joshua Oluwasegun Jatto

Department of Computer Science, University of Port Harcourt, Port Harcourt, Nigeria
Email: linda.oghenekaro@uniport.edu.ng

How to cite this paper: Oghenekaro, L.U., Olughu, I.E. and Jatto, J.O. (2023) Enhanced Document Retrieval System Using Suffix Tree Clustering Algorithm. *Open Access Library Journal*, **10**: e10228.
https://doi.org/10.4236/oalib.1110228

Received: May 9, 2023
Accepted: July 17, 2023
Published: July 20, 2023



Abstract

Most search engines in use today present the user with a single-ordered list of documents matching the search query leading to lexical ambiguity. An alternative to a single-ordered list is to cluster the search results and present a list of clusters to the user. This study implements the suffix tree clustering algorithm to optimize search. The user selects which cluster appears most relevant and the search results in that cluster are then displayed in a list under the assumption that similar documents are likely to be relevant to the same query. The proposed system clusters search results from the file system. The proposed system allows the user to issue a search query and we return the results as a set of coherent clusters. The suffix tree clustering algorithm efficiently determined documents that share common phrases. The nodes in the suffix tree define the initial cluster and to increase the number of documents in each cluster, clusters that are sufficiently similar are merged. The proposed system adopted web technologies such as hypertext markup language (HTML), and cascade styling sheet (CSS), to design the interface, while Javascript programming language was used to implement the entire system. The proposed system was implemented using PHP5 and MySQL database. The experimental results show that the suffix tree clustering algorithm can be used to cluster documents efficiently. The resulting system demonstrated an optimized search of 4.1 trillion search results of the word "Electricity" whereas a total result of 4.3 trillion was retrieved by the conventional Google Search Engine.

Subject Areas

Computer Science

Keywords

Retrieval System, Document, Clustering Algorithm, Suffix Tree, Document Clustering

1. Introduction

The need to understand large, complex, information-rich data sets is common in virtually all fields of business, science, and engineering. The ability to extract useful knowledge hidden in these data and to act on that knowledge is becoming increasingly important in everyday people's life and work. Data clustering is an unsupervised data mining approach that groups a large collection of objects (data points, records, documents etc.) into more meaningful smaller subgroups [1] [2].

Clustering is a division of data into groups of similar objects where each group, called cluster, consists of objects that are similar between themselves and dissimilar to objects of other groups. In other words, the goal of a good document clustering scheme is to minimize intra cluster distances between documents, while maximizing inter-cluster distances using an appropriate distance measure between documents. Clustering itself is not one specific algorithm, but the general task to be solved [1]. A distance measure or, dually, similarity measure thus lies at the heart of document clustering. Clustering is the most common form of unsupervised learning and this is the major difference between clustering and classification. No super vision means that there is no human expert who has assigned documents to classes. In clustering, it is the distribution and makeup of the data that will determine cluster membership [3].

Clustering can be considered the most important unsupervised learning problem, understanding large and complex information data sets is commonly seen almost in all fields of human endeavour. The ability to remove useful hidden knowledge in a data and to act on it is becoming an important issue virtually in all works of life. Using a computer-based algorithm for discovering knowledge from hidden data set in data mining is also an iterative method within which progress is defined through automatic or manual process. It will often be necessary to modify data preprocessing and model parameters until the result achieves the desired properties [4]. There are many natural ways of obtaining user information and preference using algorithms like suffix tree that can use both online learning through user interactions with the search engine during document search. With the relevance feedback of the interaction between search engine and the user at large based on the user's analysis, decisions and preferences according to [5] [6].

A suffix tree is a data structure that admits efficient string matching and querying suffix trees have been studied and used extensively and have been applied to fundamental sting problems such as finding the longest repeated substring, strings comparing and text compression. The suffix trees became useful as the search time is independent of the length of the string [7] [8].

Document clustering has been investigated for use in a number of different areas of text-mining and information retrieval. Initially, document clustering was investigated for improving the precision or recall in information retrieval systems and as an efficient way of finding the nearest neighbours of a document

More recently, clustering has been proposed for use in browsing a collection of documents or in organizing the results returned by a search engine in response to a user's query [4] [9] [10]. Clustering is a common form of unsupervised learning and is widely used in many areas of research and science. According to [4] the following are the basic directions in which clustering is used, finding similar documents, organizing large document collections, Duplicate content detection, recommendation system, search optimization.

Clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters). It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition image analysis, information retrieval, and bio-informatics. Clustering itself is not one specific algorithm, but the general task to be solved [11]. Clustering can therefore be formulated as a multi-objective optimization problem. The appropriate clustering algorithm and parameter settings (including values such as the distance function to use, a density threshold or the number of expected clusters) depend on the individual data set and intended use of the results. Clustering as such is not an automatic task, but an iterative process of knowledge discovery or interactive multi objective optimization that involves trial and failure. It will often be necessary to modify data preprocessing and model parameters until the result achieves the desired properties [4].

More recently, a new model for document representation has been introduced, based on suffix tree, which is called Suffix Tree Document Model. This model is utilized to cluster web-documents in [12]. A phrase is an ordered sequence of words, which captures more semantic of text as compared to a single word. Hence, the clustering results produced by phrase-based similarity measure are of high quality when compared to the semantic interpretation of the corpus. The more recent work on the phrase-based approach is in [11]. A similar work that also utilizes suffix tree model is from [6].

This algorithm starts by computing frequent two-word sets based on user-specified minimum support. Next, all words not in any of the frequent two-word sets are removed from the documents, resulting in compact representation of documents. The compact documents are added one-by-one into a generalized suffix tree data structure. The algorithm then traverses the generalized suffix tree in a depth-first fashion. It offers the possibilities like: grouping similar results [7] comprehend the link between the results [8] and creating the succinct representation and display of search results.

2. Methodology

The proposed suffix tree clustering algorithm designed to meet the requirements of post-retrieval clustering of web search results. Suffix tree clustering is unique in treating a document as a string, not simply a set of words, thus making use of

proximity information between words. Suffix tree clustering relies on a suffix tree to efficiently identify sets of documents that share common phrases and uses this information to create clustering and to succinctly summarize their contents for users.

The design of this system adopts the structured system analysis and design methodology (SSADM). The Structured System Analysis and Design Methodology is a system approach to the analysis and design of a property which has three design modelling known as:

1) Logical Data Modelling: this is the process of identifying, modelling and documenting the data requirements of the system being designed. The data are separated into entities (things about which a business needs to record information) and relationships (the associations between the entities).

2) Data Flow Modelling: This is a process of identifying, modelling and documenting how data moves around an information system. Data Flow Model examines processes (activities that transform data from one form to another), data stores (the holding areas for data), external entities (what sends data into a system or receives data from a system), and data flows (routes by which data can flow).

3) Entity Behaviour Modelling: this is the process of identifying, modelling arid documenting the events that affect each entity and the sequence in which these events occur.

In the design of this system, we used the suffix tree clustering algorithm to cluster a corpus of documents. The system of the project comprises of system flowchart, system architecture and system block.

2.1. System Flowchart

The system flowchart shows the flow of the activities from the start point of when the search word is put into the system, to the end point of when closely related document clusters are produced as output. The system flowchart as seen in **Figure 1**, illustrates how string are searched; the user inputs the search text which they have in mind. And the process of inputting cleans the hypertext markup language (html) to plain text. This process then identifies the base cluster by grouping the suffix of the inputted search which now deploys the output of the search words after proper clustering.

2.2. Proposed System Architecture

This is the exceptional model that defines the structure, model, behavior and more views on the proposed system (**Figure 2**). The enter search strings column allows the user to input their desired text or word. The search system which applies suffix tree in order to get the clustered search, which is being stored in the database and sent to the web administration and returns back to the database after the searched words had been clustered and then deploys to the output as searched results.

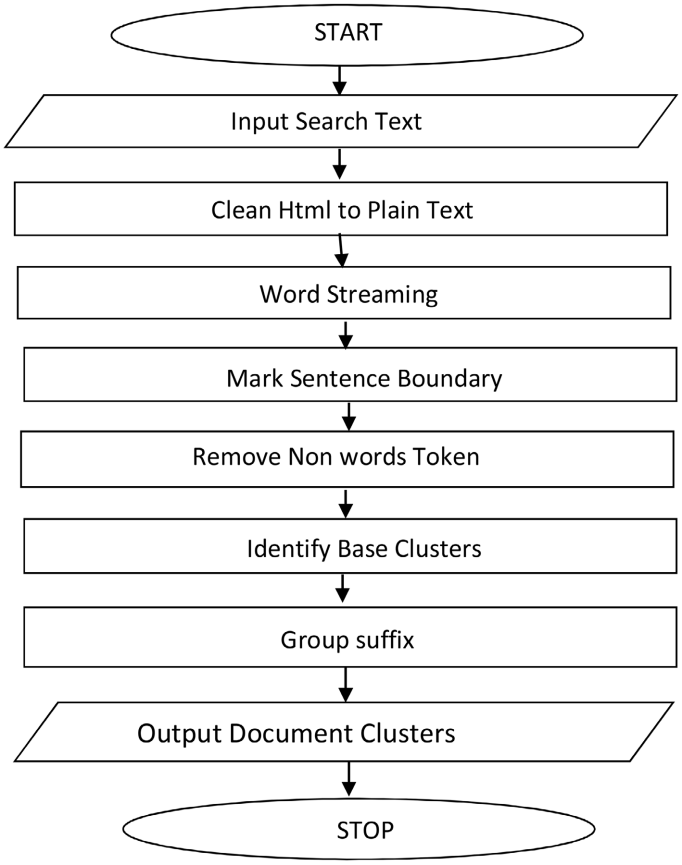


Figure 1. Proposed system flowchart.

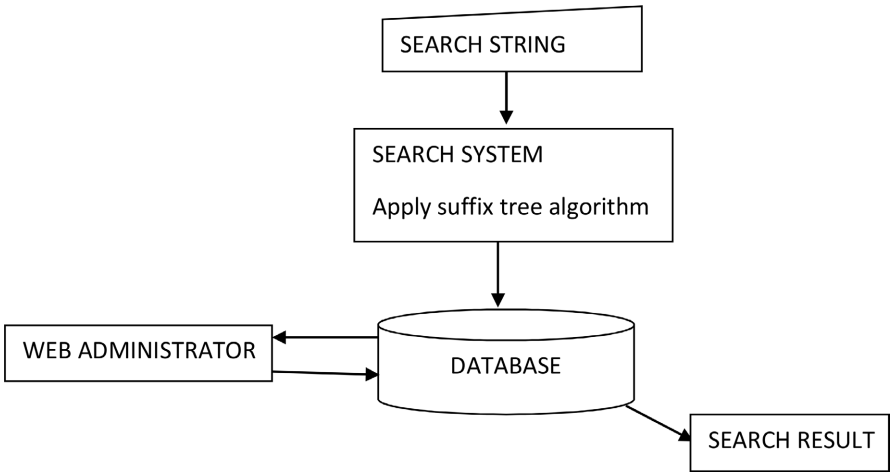


Figure 2. Proposed system architecture.

2.3. System Block Diagram

The system block diagram (**Figure 3**) refers to the high-level diagram of the system, it shows the detailed description aimed at understanding the overall concept and also for understanding the details of the implementation of the new system. The web browser helps to enter the search query, which results would be displayed, click and view the results of the searched query.

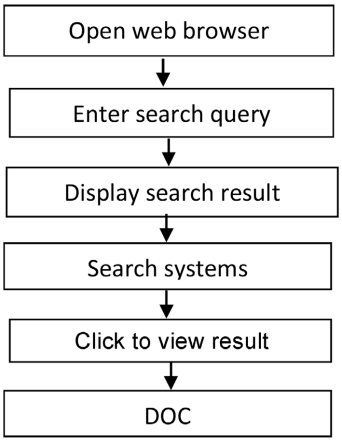


Figure 3. System block diagram of the proposed system.

2.4. Suffix Tree Algorithm

Phase 1	Scan the *String* and partition *Suffixes* based on the first *prefixlen* symbols of each suffix
Phase 2	Do for each partition:
1	START BuildSuffixTree
2	Populate *Suffixes* from current partition
3	Sort *Suffixes* on first symbol using *Temp*
4	Output branching and leaf nodes to the Tree
5	Push the nodes pointing to an unevaluated range onto the *Stack* While *Stack* is not empty
6	Pop a node
7	Find the Longest Common Prefix (LCP) of all the suffixes in this range by checking the *String*
8	Sort the range in *Suffixes* on the first symbol using *Temp*
9	Write out branching nodes or leaf nodes to *Tree*
10	Push the nodes pointing to an unevaluated range onto the *Stack*
11	END

So when the results are being displayed in the search Engine, the results related to "data", "mining", or some "data … (any strings between two) … mining" are also displayed. This can also enable an easy way to even cluster the results into related phrases [11] [12].

2.5. Suffix Tree Clustering (STC)

Apart from the introduction of Suffix tree, it is very necessary to relate Suffix tree to the clustering technique which the paper intended to address. The STC algorithm involves five steps [11] [12]:

1) Preparation of the documents: this involves retrieving document snippets from Google and results stemming.

2) Construction of Suffix tree: this is the process of inserting the string associated with the document in the Suffix tree.

3) Clustering Merging: this include the combination of two or more similar nodes of Suffix tree.

4) Clustering labeling: each cluster should have a label.

5) Clusters scoring: this has to do with clusters ranking.

2.6. Input Design

Input to the suffix tree clustering for document retrieval is a search string to be retrieval from a set of documents. A prototype of the user interface is shown in Figure 4, and it shows the input design in block diagram, where users input their queries and a search box where the users click to process the queries.

2.7. Output Design

The output of this system is a cluster of documents that contains possible documents that contains the search string. The side bars which display various forms of information to the right and left side of the system (Figure 5).

3. Result and Discussion

The result of a suffix tree clustering algorithm can be discussed in terms of its

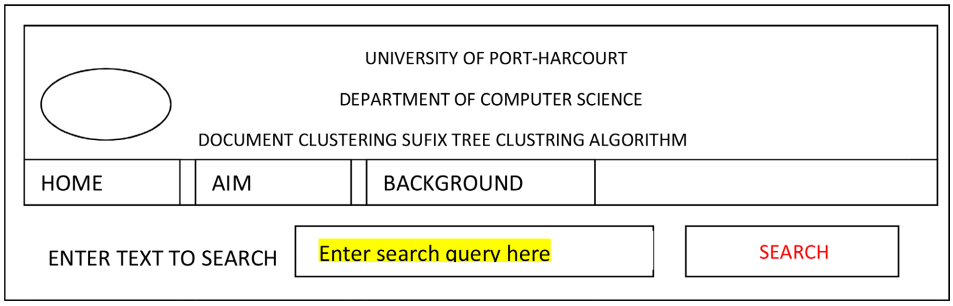


Figure 4. Input design of the proposed system.

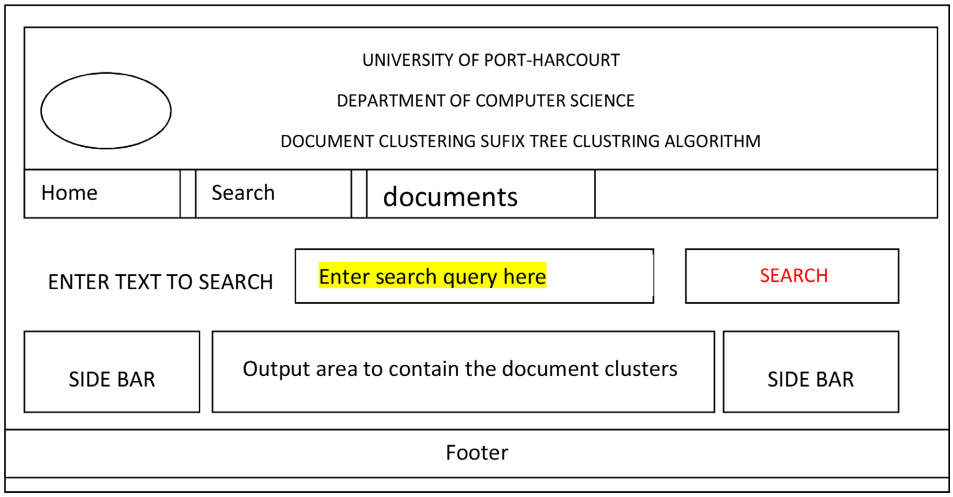


Figure 5. Output design of the proposed system.

accuracy, efficiency, and scalability. Using the searched word Hospital.

1) Accuracy: the accuracy of the algorithm can be evaluated based on how well it groups similar data points and separates dissimilar data points. When the word Hospital was searched, the STC search engine made sure to filter the relevant or the exact results of the searched word.

2) Efficiency: the time and memory complexity of the algorithm can be analysed to determine its efficiency, especially in handling large data sets. The STC search engine ensured that relevant or exact result was presenting thereby reducing time and memory.

3) Scalability: the ability of the algorithm to handle an increase in the size of the data set can also be evaluated. This includes its ability to process data in parallel or distributed manner.

Suffix tree clustering search engines differ from conventional search engines in several ways.

1) Indexing: conventional search engines use indexes for indexing, while suffix tree clustering search engines use suffix tree for indexing.

2) Search Efficiency: conventional search engines use algorithm such as Boolean search and vector space model to search, while suffix tree clustering search engines use clustering algorithms to search, which are often more efficient.

3) Query Handling: conventional search engines use simple keyword-based queries, while suffix tree clustering search engines can handle more complex queries such as regular expressions.

4) Result Relevance: conventional search engines rely on ranking algorithm and external signals such as page authority to determine result relevance, while suffix tree clustering search engines rely on the similarity of the query and documents in the clusters.

5) Scalability: conventional search engines may struggle with scaling as the number of documents increases, while suffix tree clustering search engines can handle very large datasets more efficiently.

Figure 6 and Figure 7 shows the comparison between STC search engines and Conventional search engine using the search word "electricity". In Figure 6 there were about 4,320,000,000 searched results, while in Figure 7 shows how the STC search engine clustered the query and reduced the result of the searched word to 4,010,000,000. In this observation the STC search engine had fewer, relevant, and exact results whereas the conventional search engine had many results with less relevance.

4. Conclusion

The proposed suffix tree clustering (STC) algorithm is for clustering documents. STC is a linear time clustering algorithm that is based on a suffix tree which efficiently identifies sets of documents that share common phrases. It treats a document as a string, making use of proximity information between words. It is

Figure 6. STC search engine.

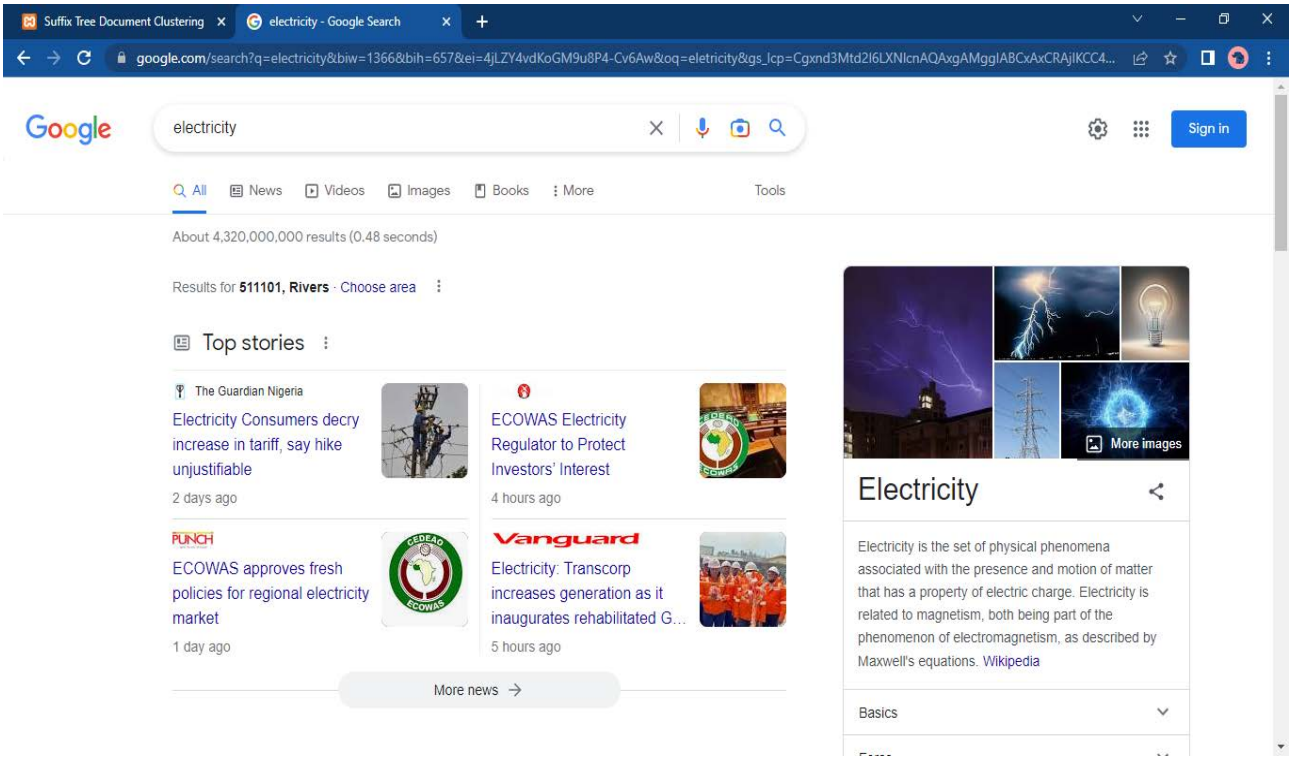


Figure 7. Conventional Search Engine

novel, incremental and 0(n)-time algorithm. STC succinctly summarizes clusters' contents for users. Therefore, suffix tree clustering is an effective and efficient technique for summarizing large volume of text data. Its implementation

involves several steps and can benefit from advances in algorithms and techniques for string processing and document clustering.

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] Chim, H. and Deng, X. (2018) Efficient Phrase-Based Document Similarity for Clustering. *IEEE Transaction on Knowledge and Data Engineering*, **20**, 1217-1229. https://doi.org/10.1109/TKDE.2008.50

[2] Chung, S.M., Holt, J.D. and Li, Y. (2014) Text Document Clustering Based on Frequent Word Meaning Sequences. *Data &Knowledge Engineering*, **64**, 381-404. https://doi.org/10.1016/j.datak.2007.08.001

[3] Hill, D.R. (2016) A Vector Clustering Technique. In: Samuelson, K., Ed., *Mechanized Information Storage*, *Retrieval and Dissemination*, North Holland, Amsterdam.

[4] Campi, A. and Ronchi, S. (2019) The Role of Clustering in Search Computing. 2009 20*th International Workshop on Database and Expert Systems Application*, Linz, 31 August-4 September 2009, 432-436.

[5] Baeza-Yates, R.A. and Gonnet, G.H. (2021) Fast Text Searching for Regular Expressions or Automaton Searching on Tries. *Journal of the ACM*, **43**, 915-936. https://doi.org/10.1145/235809.235810

[6] Farach-Colton, M., Ferragina, P. and Muthukrishnan, S. (2010) On the Sorting-Complexity of Suffix Tree Construction. *Journal of the ACM*, **47**, 987-1011. https://doi.org/10.1145/355541.355547

[7] Giegerich, R. and Kurtz, S. (2016) From Ukkonen to MeCreight and Weiner: A Unifying View of Linear-Time Suffix Tree Construction. *Algorithmica*, **19**, 331-353.

[8] Porter, M.F. (2022) An Algorithm for Suffix Stripping. *Program*: *Electronic Library and Information Systems*, **14**, 130-137. https://doi.org/10.1108/eb046814

[9] Hammouda, K.M. and Kamel, M.S. (2015) Efficient Phrase-Based Document Indexing for Web Document Clustering. *IEEE Transaction on Knowledge and Data Engineering*, **16**, 1279-1296. https://doi.org/10.1109/TKDE.2004.58

[10] Roccbio, J.J. (2019) Document Retrieval Systems—Optimization and Evaluation. Ph.D. Thesis, Harvard University, Boston.

[11] Cutting, D.R., Karger, D.R, Pedersen, J.O. and Tukey, J.W. (2015) Scatter/Gather: A Cluster-Based Approach to Browsing Large Document Collections. *Proceedings of the* 15*th International ACM SIGIR Conference on Research and Development in information Retrieval*, Copenhagen, 21-24 June 1992, 318-329.

[12] Oren, Z. and Oren, E. (1999) Grouper: A Dynamic Clustering Interface to Web Search Results. *Computer Networks*, **31**, 1361-1374.