*IOSR Journal of Computer Engineering (IOSR-JCE)*
*ISSN: 2278-0661, ISBN: 2278-8727, PP: 50-57*
*www.iosrjournals.org*

# Clustering Web Search Results-A Review

Mrs.D.A.Nikam1,Mr.N.P.Jadhav2, Miss.A.B.Shikalgar3,Mr.P.A.Chougule4

[1](Computer Science and Engineering, Dr.J.J.Magdum College of Engineering, Jaysingpur., India)
2(Computer Science and Engineering, Dr.J.J.Magdum College of Engineering, Jaysingpur., India)
3(Computer Science and Engineering, Dr.J.J.Magdum College of Engineering, Jaysingpur., India)
4(Computer Science and Engineering, Dr.J.J.Magdum College of Engineering, Jaysingpur., India)

***ABSTRACT***:*The rapid growth of the Internet has made the Web a popular place for collecting information. Today, Internet user access billions of web pages online using search engines. Information in the Web comes from many sources, including websites of companies, organizations, communications and personal homepages, etc. Effective representation of Web search results remains an open problem in the Information Retrieval (IR) community. Web search result clustering has been emerged as a method which overcomes these drawbacks of conventional information retrieval (IR) community. It is the clustering of results returned by the search engines into meaningful, thematic groups. This paper gives issues that must be addressed in the development of a Web clustering engine and categorizes various techniques that have been used in clustering of web search results. Search results clustering, the core of the system, has specific requirements that cannot be addressed by classical clustering algorithms. We emphasize the role played by the quality of the cluster labels as opposed to optimizing only the clustering structure.*

***Keywords –*** *Web search, clustering, information retrieval*

## 1. INTRODUCTION

Clustering of Web search results has been in the focus of the information retrieval community since the early days of the Web. There are two reasons for clustering of search results. The first is that the IR research community has long recognized the validity of the clustering approach in top-ranked documents, i.e. similar documents tend to be relevant to the same request. A second reason is that the ranked list is usually too large and contains many documents that are irrelevant to the particular meaning of the query the user had in mind. Thus, it would be beneficial to group search results by various meanings of the query. The attempts have made the clustering of search results for many web users. [1] 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. There are many web clustering engines available on the web such asCarrot2, Vivisimo, SnakeT, Grouper etc which give the search results in forms of clusters. A web clustering engine takes the result, returned by the search engine as input and performs clustering and labeling on that result. This process is usually seen as complementary rather than alternative and different 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. Having divided the result set into clusters, the user can quickly narrow down his search further by selecting a cluster. This resembles query refinement, but avoids the need to query the search engine for each step. 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 [3] is held as the predecessor and conceptual father of all web search result clustering. Web Search engine is the most commonly used tool for information retrieval on the web; however, its current status is far from satisfaction for several possible reasons [4], such as different users have different requirements and expectations for search results; sometimes queries cannot be expressed clearly just in several keywords; Synonymous and polysemous words make searching more complicated etc.

Figure 1, a snapshot of the Cluster based web search clearly shows how an ambiguous word like –mouse‖ relates to different groups. From the left pane of the engine, a user can easily locate what he/she is actually searching, whether computer mouse or mammal. For the same query, if a user wants to search on Yippy (formerly known as Clusty) clustering engine presents that to the user different clusters. For the same query in a traditional search engine environment, Mickey Mouse or Gaming may not appear on the first result page. Although Carrot2 and other clustering engines does far better than the average search engine, still there is a need for an efficient and effective clustering engine which is cost effective in terms of time as compared to traditional

search engines.

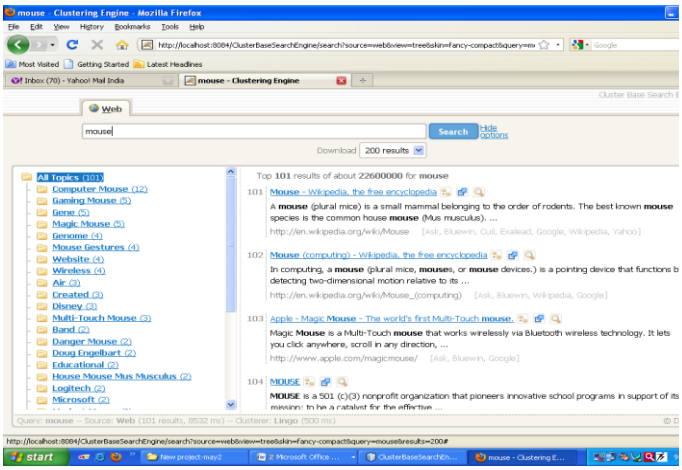

Fig 1: Cluster based web search

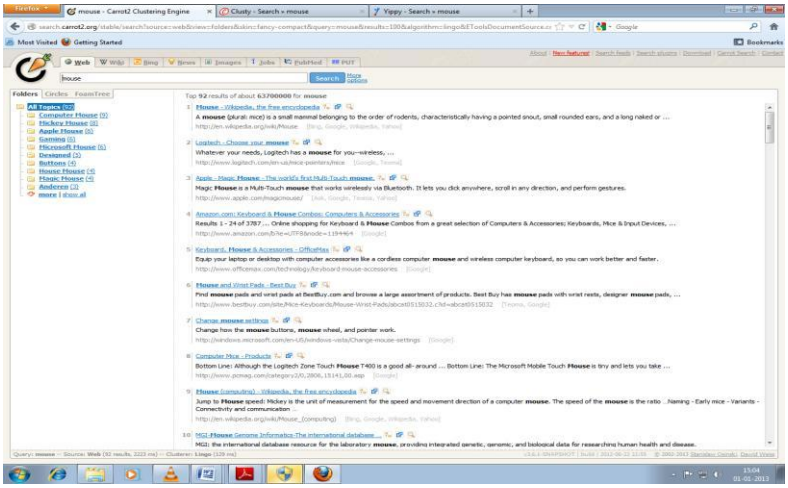

Fig 2:Carrot2 Clustering Engine

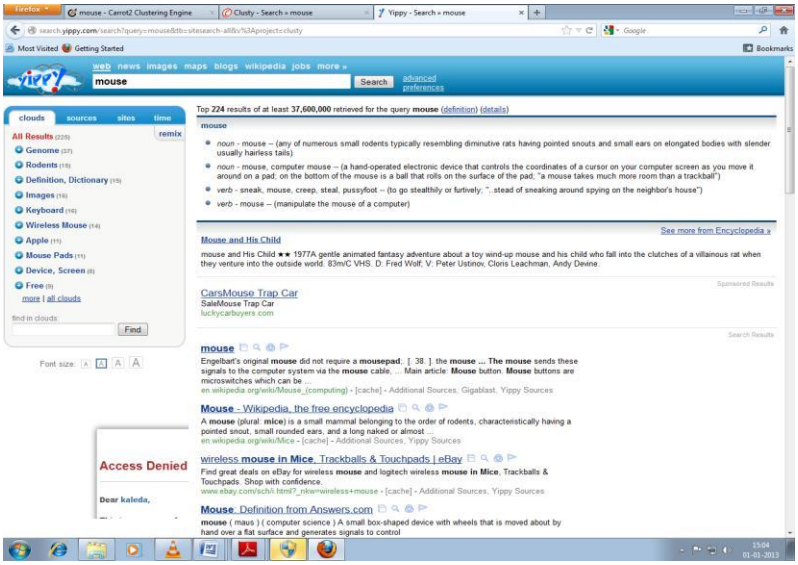

Fig 3:Yippy Clustering Engine

## 2. OVERVIEW OF WEB SEARCH CLUSTERING

The goal of clustering search result is to give user an idea of what the result contains. This idea is in the form of clusters. Clustering in context of web search result means organizing query result pages into groups based on their similarity between each other.

For cluster-based Web information retrieval it is more convenient to follow a two stage approach, in which clustering is performed as a post processing step on the set of documents retrieved by an information retrieval system on a query. Post retrieval clustering is not only more efficient than pre retrieval clustering, but it may also be more effective. The reason is that pre retrieval clustering might be based on features that are frequent in the collection but irrelevant for the query at hand, whereas post retrieval makes use only of query-specific features. There are two types of post retrieval clustering. The clustering system groups the ranked results and gives the user the ability to choose the groups of interest in an interactive manner [Allen et al. 1993;Hearst and Pedersen 1996].

A clustering engine follows the latter approach, and the expression *search results clustering* usually refers to browsing a clustered collection of search results returned by a conventional Web search engine. Search results clustering can thus be seen as a particular subfield of clustering concerned with the identification of thematic groups of items in Web search results. The input and output of a search results clustering algorithm can be characterized more precisely in the following way.

The *input* is a set of search results obtained in response to a user query, each described by a URL, a title, and a snippet (a short text summarizing the context in which the query words appear in the result page). Assuming that there exists a logical topic structure in the result set, the *output* is a set of labelled clusters representing it in the closest possible way and organized in a set of flat partitions, hierarchy or other graph structure. The dynamic nature of the data together with the interactive use of clustered results poses new requirements and challenges to clustering technology, as detailed in the following list.

*1. Meaningful labels*: - Traditional algorithms use the cluster centroid as cluster representative, which is of little use for guiding the user to locate the sought items. Each cluster label should concisely indicate the contents of the cluster items, while being consistent with the meanings of the labels of more general and more specific clusters.

2. *Computational efficiency:* - Search results clustering are performed online, within an acquisition of search results, whereas the efficiency of the cluster construction algorithm is less important due to the low number of input results.

3. *Short input data description*: - Due to computational reasons, the data available to the clustering algorithm for each search result are usually limited to a URL, an optional title, and a short excerpt of the document's text (the snippet). This contrasts with using more coherent, multidimensional data descriptions such as whole Web pages.

4. *Unknown number of clusters*: - Many traditional methods require this number as an input. In search results clustering, however, both the number and the size of clusters cannot be predetermined because they vary with the query; furthermore, they may have to be inferred from a variable number of search results.

5. *Overlapping clusters:* - As the same result may often be assigned to multiple themes, it is preferable to cater for overlapping clusters. Also, a graph may be more flexible than a tree because the latter does not easily permit recovery from bad decisions while traversing the cluster structure. Using a graph, the results contained in one cluster can be reached through several paths, each fitting a particular user's need or paradigm.

6. *Graphical user interface (GUI):*- A clustering engine allows interactive browsing through clustered Web search results for a large population of users. It can thus take advantage of Web-based graphical interfaces to convey more visual information about the clusters and their relationships, provided that they are intuitive to use and computationally efficient.

## 3. ARCHITECTURE AND TECHNIQUES OF WEB SEARCH CLUSTERING

Practical implementations of clustering Web search results usually consist of four general components: Retrieval of search results, preprocessing of search results, cluster creation and labeling, and visualization of clustered results, all arranged in a processing pipeline shown in Figure 4.

3.1 Retrieval of search results-

The task of the retrieval of search results is to provide input for the rest of the system. Based on the user-specified query string, the retrieval component must deliver typically between 50 and 500 search results, each of which should contain a title, a contextual snippet, and the URL pointing to the full text document being referred to.

A potential source of search results for the retrieval component is a public Web search engine, such as Yahoo!, Google, or Live Search. The most elegant way of fetching results from such search engines is by using application programming interfaces (APIs) these engines provide. In the case of Yahoo!, for example, search

results can be retrieved in a convenient XML format using a plain HTTP request . At the time of writing, all major search engines (Google, Yahoo!, Live Search) provide public APIs, but they come with certain restrictions and limitations. These restrictions are technical (maximum number of queries per day, maximum number of fetched results per query), but also legal—terms of service allowing only non-commercial or research use. While the former can affect the overall performance of a clustering engine, the latter involves a risk of litigation.

Another way of obtaining results from a public search engine is called *HTML scraping*.The search results retrieval component can use regular expressions or other form of mark up detection to extract titles, snippets, and URLs from the HTML stream served by the search engine to its end users. Due to relatively frequent changes of the HTML mark up, this method, if done manually, would be rather tedious and lead to reliability and maintenance problems. HTML scrapers can be trained using machine learning methods to extract the content automatically, but such techniques are not widespread. Additionally, even though this technique was popular in the early days of Web search clustering engines, it is nowadays strongly discouraged as it infringes search engines legal terms of use. Besides, the public APIs we mentioned provide a more reliable and faster alternative. There exists an encouraging movement to standardize access to search engines called Open Search (www.opensearch.org). A number of search feeds from various sources are already available, but unfortunately the ones from major vendors return results in HTML format only (suitable for scraping, but not for automated processing).An alternative source of search results for the retrieval component is a dedicated document index. This scenario is particularly useful when there is a need to search and cluster documents that are not available through public search engines, for example, enterprise, domain-specific content or IR test collections. In this case, the retrieval component may additionally need to take responsibility for generating contextual document snippets.

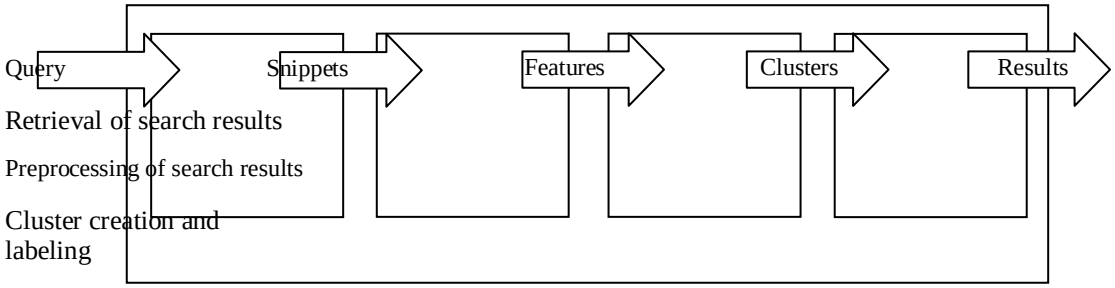


Fig 4: components of Clustering web search results

3.2 Preprocessing of search results :-

Input preprocessing is a step that is common to all search results clustering systems. Its primary aim is to convert the output by the retrieval component into a sequence of *features* used by the actual clustering algorithm. A flow goes through language identification, tokenization, and shallow language preprocessing (up to stemming and stop word removal) and finally selection of features.

Clustering engines that support multilingual content must perform initial *language recognition* on each search result in the input. Information about the language of each entry in the search result is required to choose a corresponding variant of subsequent components—tokenization and stemming algorithms—but it also helps during the feature selection phase by providing clues about common, unimportant words for each language (stop words) that can be ignored. Another challenge for language identification in search results clustering is the small length of input snippets provided for clustering.

Finally, when binding a Web clustering engine to a source of search results that can expose documents as sequences of tokens rather than raw text, the tokenization step can be omitted in the clustering algorithms processing pipeline. Section 5 provides more implementation-level details on such a setting.

*Stemming* is a typical shallow NLP technique. The aim of stemming is to remove the inflectional prefixes and suffixes of each word and thus reduce different grammatical forms of the word to a common base form called a *stem*. For example, the words *connected*, *connecting*, and *interconnection* would be transformed to the word *connect*. Note that while in this example all words transform to a single stem, which is also a dictionary word, this may not always be the case—a stem may not be a correct word. In fact it may not be a word at all—a stem is simply a unique token representing a certain set of words with roughly equal meaning. Because of this departure from real linguistics, stemming is considered a heuristic method and the entire process is dubbed

*shallow* linguistic preprocessing . The most commonly used stemming algorithm for English is the Porter stemmer [5]. When a great deal of input text is available, stemming does not seem to help much. On the other hand, it plays a crucial role when dealing with very limited content, such as search results, written in highly inflectional languages.

Last but not least, the preprocessing step needs to extract features for each search result present in the input. In general data mining terms, features are atomic entities by which we can describe an object and represent its most important characteristic to an algorithm. When looking at text, the most intuitive set of features would be simply words of a given language, but this is not the only possibility. A feature class with numerous possible values (like all words in a language) is often impractical since certain elements are closely related to each other and can form equivalent classes (for example all words with a unique stem), and other elements occur very infrequently or not at all. Many features are also irrelevant for the task of assessing similarity or dissimilarity between documents, and can be discarded. This is where feature extraction, selection and construction takes place. These techniques are well covered by a number of books and surveys (Yang and Pedersen [6] and Liu et al. [7] place particular emphasis on text processing),and we will omit their detailed description here, stopping at the conclusion that the choice of words, albeit the most common in text processing, does not exhaust the possibilities.

Regardless of what is extracted, the preprocessing phase ends with some information required to build the *model* of text representation—a model that is suitable for the task of document clustering and indirectly affects a difficult step that takes place later on—cluster labeling.

### 3.3 Cluster creation and Labeling

The set of search results along with their features, extracted in the preprocessing step, are given as input to the clustering algorithm, which is responsible for building the clusters and labelling them. Because a large variety of search results clustering algorithms have been proposed, this raises the question of their classification. Clustering algorithms are usually classified according to the characteristics of their output structure, but here we take a different viewpoint.

In search results clustering, users are the ultimate consumers of clusters. A cluster which is labelled awkwardly, ambiguously, or nonsensically is very likely to be entirely omitted even if it points to a group of strongly related and somehow relevant documents. This observation led to a very important conclusion (credited to Vivisimo) in search results clustering description comes first.

### 3.4 Visualization of clustered results

As the clustered results are used for browsing retrieval, the scheme chosen for visualizing and manipulating the hierarchy is very important. The largely predominant approach is based on hierarchical folders shown in Figure 1and Figure 2.The system displays the labels of the top level of the cluster hierarchy using and the user may click on each label to see the web pages associated with it as well as to expand its sub clusters, if any. The user can then repeat the same operation on the newly displayed sub clusters.

Usually, the most populated clusters are shown earlier and the clusters with very few snippets are displayed on demand. Furthermore, all the snippets of one cluster that are not covered by its displayed children are grouped into a cluster named ―other.‖

Figure 1 is an example in which only the top clusters of the top level of the hierarchy are shown, and in which the user chose to see the documents associated with the cluster ―mouse‖ without expanding its subclusters.The folder-tree display has been successfully adopted by Vivisimo and by many other systems, including Carrot2, CREDO, and SnakeT. As the hierarchical folders are used for storing and retrieving files, bookmarks, menus items, and so on, most users are familiar with it and hence no training is required.

Through a graphical display, the relationships in size, distance, and kind (folder or search results) between the clusters can be rendered by rich spatial properties such as dimension, color, shape, orientation, enclosure, and adjacency. The research on information visualization has explored a huge number of techniques for representing trees. Two excellent reviews of this work are Herman et al. [8] and Katifori et al. [9], the latter with a focus on information retrieval.

In practice however, only a few alternative tree visualization schemes to the folder layout have been used in the deployed clustering engines. The most notable system is Carrot2 (circles and foam tree visualization). Taking a nesting and zooming approach, Carrot2 displays each cluster as a circle and when you clisk on that it shows documents related to that cluster (see Figure 5). The system allows the user to zoom in on the child nodes, making them the current viewing level.

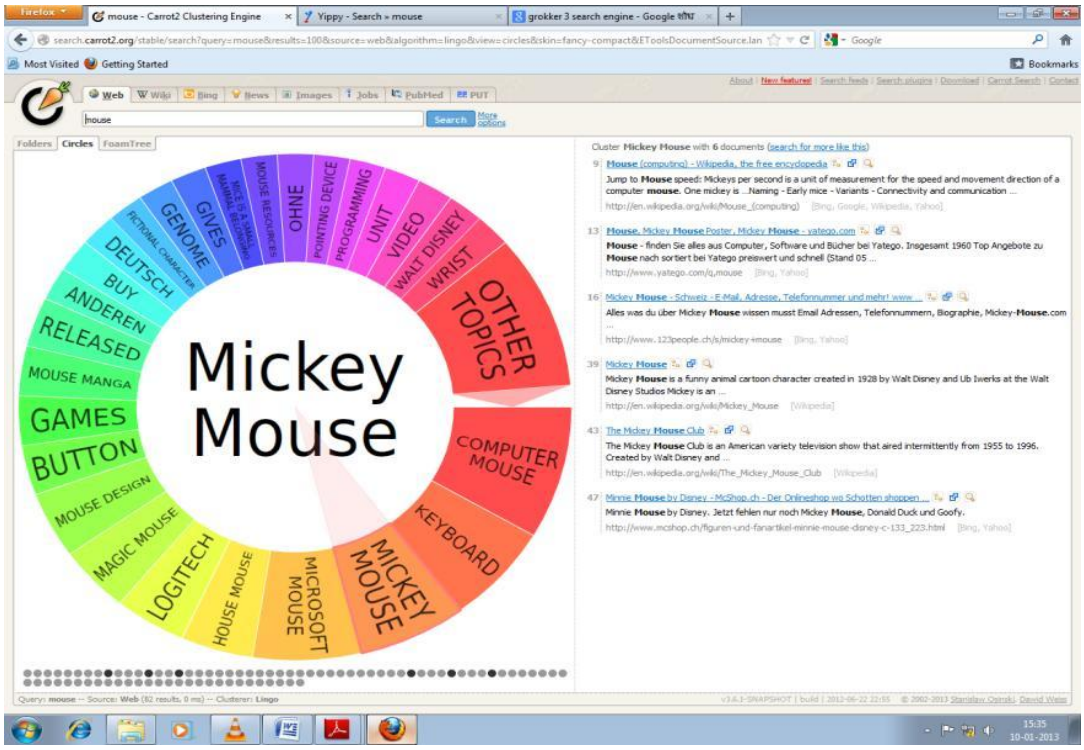


Fig 5: circle visualization for query –mouse‖

## 4. REVIEW OF WEB CLUSTERING ENGINES

In the section, we briefly review the most influential search results clustering systems that have appeared in literature or have made a commercial impact. We make a distinction between research and commercial systems, even if such a separation may sometimes be difficult. Descriptions of research systems are shown in Table1, commercial systems are ordered alphabetically and shown in Table2. Neither list is exhaustive.

| System/Algorithm Name | Cluster labels | Clustering methods | Clusters structure |
|---|---|---|---|
| Grouper (STC) | phrases | STC | flat, Concept cloud |
| Lassi | lexical | AHC | hierarchy |
| CIIRarchies | word sets | language model/graph analysis | hierarchy |
| WICE (SHOC) | phrases | SHOC | hierarchy |
| Carrot2 (Lingo) | phrases | Lingo | flat |
| Carrot2 (TRSC) | n-grams | TRSC | flat |
| WebCat | words | K-Means | flat |
| AISearch | word sets | AHC | hierarchy |
| CREDO | word sets | concept lattice | graph |
| DisCover | phrases | incremental coverage optimization | hierarchy |
| SnakeT | phrases | Approx. sent. Coverage | hierarchy |
| SRC | n-grams | SRC | flat |
| EigenCluster | single words | divide-merge | flat |
| WhatsOnWeb | single words | Edge connectivity | graph |

Table 1. Summary of Research Search Results Clustering systems

| Name | Company | Cluster labels | URL | Clusters structure |
|---|---|---|---|---|
| Accumo | Accumo | Phrases | www.accumo.com | Tree |
| Clusterizer | CyberTavern | Phrases | www.iboogie.com | Tree |
| Cowskid | Compara | Terms | www.cowskid.com | Flat |

| | | | | |
|---|---|---|---|---|
| Fluster | Funnelback | Phrases | www.funnelback.com | Flat |
| Grokker | Grokker | Phrases | www.grokker.com | Graphical/Tre |
| KartOO | KartOO | Phrases | www.kartoo.com | Graphical/Tre |
| Lingo3G | Carrot Search | Phrases | www.carrot-search.com | Tree |
| Mooter | Mooter Media | Phrases | www.mooter.com | Graphical/Flat |
| WebClust | WebClust | Phrases | www.webclust.com | Tree |
| Vivisimo | Vivisimo | Phrases | www.vivisimo.com | Tree |

Table 2. Commercial Companies Offering Technologies for Clustering Search Results

## 5. IMPLEMENTATION ISSUES

Several factors contribute to the overall computational performance of a search results clustering engine. The most critical tasks involve the first three components presented in Figure 4, namely retrieval of search result, preprocessing, and clustering. The visualization component is not likely to affect the overall system efficiency in a significant manner.

5.1 Retrieval of Search result:-No matter whether search results are fetched from a public search engine (e.g., through Yahoo! API) or from a dedicated document index, collecting input for clustering amounts to a large part of the total processing time. In the first case, the primary reason for the delay is the fact that the number of search results required for clustering (e.g., 200) cannot be fetched in one remote request. The Yahoo! API allows up to 50 search results to be retrieved in one request, while Google SOAP API returns a mere 10 results per one remote call. Issuing a number of search requests in parallel can decrease the total waiting time, the delays still fall in the 1–6 second range.

When fetching search results from a dedicated document index, the overhead related to network communication can be avoided. In this scenario, however, the document retrieval engine will need to extract the required number of documents from its internal storage, which can be equally time-consuming. Additionally, the engine may also need to create contextual document snippets, which will increase the processing time even further.

5.2 Clustering: - Depending on the specific algorithm used, the clustering phase can significantly contribute to the overall processing time. Although most of the clustering algorithms will have common components, such as input tokenization and stemming, the efficiency of clustering is mainly a function of the computational complexity of the core clustering element of a particular algorithm. The clustering component may in turn comprise several steps.

In the case of the Lingo algorithm, for example, the most time consuming operation is computing the SVD decomposition of the term-document matrix with a computational complexity equal to $O(m^2n + n^3)$, for a $m \times n$ matrix, where in the case of Lingo, n is the number of search results being clustered, and m is the number of features used for clustering. While this may seem costly, search results clustering is a very specific domain, where problem instances rarely exceed a few hundred snippets (it is impractical to fetch more than, say 500 search results, because of the network latencies). Search results clustering systems must therefore be optimized to handle smaller instances and process them as fast as possible. This assumption to some degree justifies the use of techniques that would be unacceptable in other fields due to scalability issues.

Concluding the performance discussion let us again emphasize that the efficiency of search results clustering is the sum of the time required to fetch the input from a search engine and the time to cluster this input. For the Lingo algorithm, the time spent on cluster construction accounts for about 10–30% of the total processing time (depending on whether a local index or remote data source was used).

## 6. CONCLUSION

In this article, we have presented the most important scientific and technical aspects of Web search result clustering. We have discussed the issues that must be addressed to build a Web clustering engine and have reviewed and evaluated a number of existing algorithms and systems. A number of advances must be made before a search result clustering entirely fulfills the promise of being the PageRank of the future. First, more work needs to be done to improve the quality of the cluster labels and the coherence of the cluster structure. Second, more studies on user queries must be made to understand when search results clustering is most useful. Third, there is a need for carefully engineered evaluation benchmarks to allow cross-system comparison, and to measure progress. Fourth, advanced visualization techniques might be used to provide better overviews and guide the interaction with clustered results.

## REFERENCES

[1] D.A.Nikam,A.B.Rajmane, Cluster Based Web Search, *International Journal of Advanced Research in Computer Science & Engineering, Volume 1,issue-3,May 2012.*

[2] Carpenito C,Osinski S,Romano G,and Wriss D,A Survey of Web Clustering Engines, ACM Computing Surveys,Volume41,issue 3,Article 17,2009.

[3] Cutting DR,Kager DR,Pedersen JO and Tukey JW(1992) Scatter/gather: a cluster-based approach to browsing large document collections. The 15th international ACM Sigir conference on Research and development in information retrieval.

[4] Wang Y and Kitsuregawa M (2001) Link Based Clustering of Web Search Results. In Proceedings of The Second InternationalConference on Web-Age Information Management (WAIM2001), XiAn, P.R.China, Springer-Verlag LNCS.

[5] M. F. Porter. An algorithm for suffix stripping. Program Vol. 14, no. 3, pp 130-137.

[6] YANG, Y. AND PEDERSEN, J. O.(1997) A comparative study on feature selection in text categorization.. In Proceedings of the14th International Conference on Machine Learning (ICMC). Morgan Kaufmann, San Francisco, 412–420.

[7] LIU, T., LIU, S., CHEN, Z., AND MA, W.-Y. 2003. An evaluation on feature selection for text clustering. In Proceedings of the20th International Conference on Machine Learning, August 21–24, T. Fawcett and N. Mishra, Eds. AAAI Press, 488–495.

[8] HERMAN, I., MELANCON, G., AND MARSHALL, S. M. (2000) Graph visualization and navigation in information visualization: A survey. IEEE Trans. Visual. Comput. Graph. 6, 10, 1–21.

[9] KATIFORI, A., HALATSIS, C., LEPOURAS, G., VASSILAKIS, C., AND GIANNOPOULOU, E. 2007. Ontology visualization methods a survey. ACM Comput. Surv. 39, 4, 1–43.