The Anatomy of Web Search Result Clustering and Search Engines

R.SUBHASHINI

Research Scholar, Sathyabama University,
Chennai-119, India
subhaagopi@gmail.com

V.JAWAHAR SENTHIL KUMAR

Assistant Professor, Anna University,
Chennai, India.

Abstract

World Wide Web is a very large distributed digital information space. The ability to search and retrieve information from the Web efficiently and effectively is an enabling technology for realizing its full potential. Current search tools retrieve too many documents, of which only a small fraction are relevant to the user query. Furthermore, the most relevant documents do not necessarily appear at the top of the query output order. Clustering Techniques are now being used to give a meaningful search result on web. Text document clustering has been traditionally investigated as a means of improving the performance of search engines. We present a thorough comparison of the algorithms based on the various facets of their features and functionality. Furthermore, we highlight the main characteristics of a number of existing Web clustering engines and also discuss how to evaluate their retrieval performance.

Keywords : **Information Retrieval; Web Mining; Search Engine; Document Clustering**.

1. Introduction

With the increase in information on the World Wide Web it has become difficult to find the desired information on search engines. One approach that tries to solve this problem is using clustering techniques for grouping similar documents together in order to facilitate presentation of results in more compact form and enable thematic browsing of the results set. It is used to give a meaningful search result on web. The four main criteria for creating cluster categories: Making the titles concise, accurate, distinctive, and "humanlike" -- in other words, not something that looks like it was generated by a machine. One common feature of most current clustering engines is that they do not maintain their own index of documents; similar to meta search engines, they take the search results from one or more publicly accessible search engines. The low precision of the web search engines coupled with the long ranked list presentation make it hard for users to find the information they are looking for. It takes lot of time to find the relevant information. Typical queries retrieve hundreds of documents, most of which have no relation with what the user was looking for. According to this, we considered Web-snippet clustering engine is a useful complement to the flat, ranked list of results offered by classical search engines (like Google). Web snippet (short description) clustering also known as Web Search Results Clustering is an attempt to apply the idea of clustering to snippets returned by a search engine in response to a query. Thus, it can be perceived as a way of organizing the snippets into set of meaningful thematic groups. Actually, clustering engines are usually seen as complementary instead of alternative to search engines. Not only has search results clustering attracted considerable commercial interest, but it is also an active research area, with a large number of published papers discussing specific issues and systems. Search results clustering is clearly related to the field of document clustering but it poses unique challenges concerning both the effectiveness and the efficiency of the underlying algorithms that cannot be addressed by conventional techniques. The main difference is the emphasis on the quality of cluster labels, whereas this issue was of somewhat lesser importance in earlier research on document clustering. A clustering engine tries to address the limitations of current search engines by providing clustered results as an added feature to their standard user interface and meaningful labels. This paper gives an idea about document clustering, Web Page document clustering and clustering engines.

2. Document Clustering

Clustering can be characterized as a process of discovering subsets of objects in the input (*clusters*, *groups*) in such a way that objects within a cluster are similar to each other and objects from different clusters are dissimilar from each other, usually according to some similarity measure. The text-based web document clustering approaches characterize each document according to its content, i.e. the words (or sometimes phrases) contained in it. If two documents contain many common words then they are very similar. The text-based

approaches can be further classified according to the clustering methods they used. Furthermore, according to the way a clustering algorithm handles uncertainty in terms of cluster overlapping, an algorithm can be either *crisp* (or hard), which considers non-overlapping partitions, or *fuzzy* (or soft), with which a document can be classified to more than one cluster. Most of the existing algorithms are crisp, meaning that a document either belongs to a cluster or not.

Document clustering was proposed mainly as a method of improving the effectiveness of document ranking following the hypothesis that closely associated documents will match the same requests [van Rijsbergen (1979)]. It is generally considered to be a centralized process. Examples of document clustering include web document clustering for search users.

2.1. Key Requirements

Key Requirements for Web Document Clustering As pointed out by Zamir and Etzioni (1998) the following are the key requirements for web document clustering methods.

(1) Relevance: The method ought to produce clusters that group documents relevant to the user's query.
(2) Browsable Summaries: The user needs to determine at a glance whether a cluster's contents are of interest. Ranked lists of the clusters may infact difficult to browse. Therefore the method has to provide concise and accurate descriptions of the clusters.
(3) Overlap: Since documents have multiple topics, it is important to avoid confining each document to only one cluster.
(4) Snippet-tolerance: The method ought to produce high quality clusters even when it only has access to the snippets returned by the search engines, as most users are unwilling to wait while the system downloads the original documents off the Web.
(5) Speed: A very patient user might sift through 100 documents in a ranked list presentation. Clustering on the other hand allows the user to browse several related documents. Therefore the clustering method ought to be able to cluster up to one thousand snippets in a few seconds. For the impatient user, each second counts.
(6) Incrementality: To save time, the method should start to process each snippet as soon as it is received over the Web.

2.2. Types

Based on the relation between the clusters, the clustering algorithm is classified as partitional and hierarchical
The most common partitional clustering algorithm is k-means, which relies on the idea that the center of the cluster, called *centroid*, can be a good representation of the cluster. The algorithm starts by selecting k cluster centroids. Then the cosine distance2 between each document in the collection and the centroids is calculated and the document is assigned to the cluster with the nearest centroid. After all documents have been assigned to clusters, the new cluster centroids are recalculated and the procedure runs iteratively until some criterion is met. Another approach to partitional clustering is used in the Scatter/Gather system. Scatter/Gather [cutting et al. (1992)] uses two linear-time partitional algorithms, Buckshot and Fractionation are used for the refinement of the clusters and also apply HAC to select the initial cluster centers. The idea is to use these algorithms to find the initial cluster centers and then find the clusters using the assign-to nearest approach.

Finally, the single pass method [Rasmussen (1992)] is another approach to partitional clustering which is based on the assignment of each document to the cluster with the most similar representative is above a threshold. The clusters are formed after only one pass of the data and no iteration takes place. Consequently, the order in which the documents are processed influences the clustering. The advantages of these algorithms consist in their simplicity and their low computational complexity. The disadvantage is that the clustering is rather arbitrary since it depends on many parameters, like the values of the target number of clusters, the selection of the initial cluster centroids and the order of processing the documents.

Hierarchical Methods [Voorhees, E. M (1986)] result in a tree-like representation. The clusters of documents highly similar to each other are nested within larger clusters of less similar documents. They can be agglomerative (building the tree from individual documents) or divisive (starting with the whole set and dividing it in to clusters).

Hierarchic Agglomerative Clustering Methods (HACM)

(1) Determine all inter document similarities
(2) Form a cluster from the two closest documents or clusters
(3) Redefine the similarities between the new cluster and all other documents or clusters, leaving all other similarities unchanged. This step depends on the specific method…
(4) Repeat steps 2 and 3 until all documents are in one cluster

Some HACM's are

Single Link

- The similarity between two clusters is the maximum of the similarities between all pairs of documents such that one document is in one cluster and the other document is in the other cluster
- Each cluster member will be more similar to at least one member in that same cluster than to any member of another cluster

Complete Link

- Similarity between two clusters: minimum of the similarities between all pairs of documents
- Each cluster member is more similar to the most dissimilar member of that cluster than to the most dissimilar member of any other cluster (i.e. more cohesive clusters than Single Link)

Group Average Link

- Similarity between two clusters: mean of the similarities between all pairs of documents, such that one document of the pair is in one cluster and the other document in the other cluster

Centroid/Median Methods

Each cluster as is it formed is represented by the group centroid/median. At each stage of the clustering the pair of clusters with the most similar mean centroid/median is merged. The difference between the centroid and the median is that the second is not weighted proportionally to the size of the cluster.

Suffix Tree Clustering

In this clustering the whole web document is treated as a string. The identification of base clusters is the creation of an inverted index of strings for the web document collection.

STC Algorithm

(1) Document cleaning

Delete the word prefix and suffix, reduce plural to singular. Sentence boundaries are marked and non-word tokens (such as numbers, HTML tags and most punctuation) are stripped.

(2) Identify Base Cluster.

Create an inverted index of strings from the web document collection with using a suffix tree. Each node of the suffix tree represents a group of documents and a string that is common to all of them. The label of the node represents the common string. Each node represents a base cluster.

(3) Score base clusters.

Each base cluster is assigned a score

The score formula: $S(B)=|B|*f(|P|)$

$|B|$ is the number of documents in base cluster B

$|P|$ is the number of words in string P that has a non-zero score

The function f penalizes single word, linear for string that is two to six words long. And become constant for longer string.

(4) Combine base clusters.

Based on this algorithm, Zamir and Etzioni (1999) have developed the clustering engine named Grouper.

Table 1. Comparison of Clustering Algorithms

ALGORITHM	Time Complexity	Similarity Criterion	Overlap	Advantages	Disadvantages
Single linkage	$O(n^2)$	Join clusters with most similar pair of documents	Crisp clusters	□ Sound theoretical properties □ Efficient implementations	□ Not suitable for poorly separated clusters □ Poor quality
Group Average	$O(n^2)$	Average pairwise similarity between all objects in the 2 clusters	Crisp clusters	High quality results	Expensive in large collections
Complete linkage	$O(n^2)$	Join cluster with least similar pair of documents	Crisp clusters	Good results (Voorhees alg.)	Not applicable in large datasets
Centroid/ Median HAC	$O(n^2)$	Join clusters with most similar centroids / medians	Crisp clusters		Small changes may cause large changes in the hierarchy
K-means	O(nkt) (k: initial clusters, t: iterations)	Euclidean or cosine metric	Crisp clusters	□Efficient (no sim matrix required) □Suitable for large datasets	Very sensitive to input parameters
Single Pass	O(nlogn)	If distance to closest centroid > threshold assign, else create new cluster	Crisp clusters	□Efficient □Simple	Results depend on the order of document presentation to the algorithm
Scatter/ Gather	Buckshot: O(kn) Fractionation: O(nm)	Hybrid: first partitional, then HAC	Crisp clusters	□Dynamic Clustering □Clusters presented with summaries □Fast	□Must have a very quick clustering algorithm □Focus on speed but not on accuracy
Suffix Tree Clustering	O(n)	Sim = 1 if |Bm)Bn|/|Bm| > threshold and |Bm)Bn|/|Bn| > threshold, else Sim = 0	Fuzzy clusters	□ Incremental □Captures the word sequence	□Snippets usually introduce noise □Snippets may not be a good description of a web page

Table 1. shows the comparison of various clustering algorithm. Most of them concentrate on the two most widely used approaches to text-based clustering: partitional and HAC algorithms [Willett, P (1988)]. As mentioned earlier, among the HAC methods, the single link method has the lowest complexity but gives the worst results whereas group average gives the best. In comparison to the partitional methods, the general conclusion is that the partitional algorithms [Steinbach et al. (2000)] have lower complexities than the HAC, but they don't produce high quality clusters. Indeed, the complexity of the partitional algorithms is linear to the number of documents in the collection, whereas the HAC take at least O(n2) time. But, as far as the quality of the clustering is concerned, the HAC are ranked higher. This may be due to the fact that the output of the partitional algorithms depends on many parameters (predefined number of clusters, initial cluster centers, criterion function, processing order of documents). Hierarchical algorithms are more efficient in handling noise and outliers. Another advantage of the HAC algorithms is the tree-like structure, which allows the examination of different abstraction levels. When k-means is run more than one times it may give better clusters than the HAC.

Finally, a disadvantage of the HAC algorithms, compared to partitional, is that they cannot correct the mistakes in the merges. A few words about the complexity and limitations of STC are the base cluster phase requires a time linear in the size of the documents, but hidden constants are high and the known GST implementations require a lot of memory. The merge phase is also linear in the number of base clusters. So, the overall complexity is O(n). STC does not reduce the high dimension of the text documents, hence its complexity is quite high for large text databases. And STC just performs the word form matching, which ignores the semantic and lexical relationships between words.

3. Search Engines

The Search Engine component is a part of the Information Retrieval model component. Its main responsibility is the comparison of documents based on their document models through obtaining documents similarity values. In today's search engines, Clustering of results is the next step up from ranking of documents Web search engines did not come into existence until 1994.

A search engine has four components:

- document processor indexes new documents. Indices are a mapping between words and what documents they appear in. Most engines are spider-based, so a crawl of the web for new documents and the updating of the index is automated.
- query processor inspects a user's query and translates it into something internally meaningful.
- matching function uses the above internally meaningful representation to extract documents from the index.
- ranking scheme positions the more-relevant documents on top, using some relevance measure.

We may classify web search engines according to the set of features they explore (Broder, 2002). First generation web search engines, starting in 1994 with WebCrawler and Lycos, explore on-page data (content and formatting). They support mostly informational queries. The second generation, emerging in 1998, with Google (Page Rank), uses off-page web specific data (link analysis, anchor text and click streams data) and supports both informational and navigational queries. The third generation, appearing during the first years of 2000 attempts to merge multiple sources of evidence and aims to support all kinds of queries.

Modern Web IR is a discipline which has exploited some of the classical results of Information Retrieval developing innovative models of information access. Therefore, search engines have established as a revolutionary working metaphor. If someone needs information about a book, an address, a research paper, a flight ticket, or almost any other topic, they just make a query on a search engine. In this paragraph we briefly review the architecture of a typical search engine. The architecture of a search engine is given in Figure 1. Crawlers are distributed agents which gather information from the Web. They crawl the Web graph visiting the pages according to some policies (BFS, DFS, random, topic focused, prioritized) and store the pages in a local Page Repository. From time to time, the pages are indexed and analyzed to produce a fast indexable representation of both the documents and the link structures. Both the textual and structural indexes are then analyzed to rank the documents stored in the repository. For efficiency reasons, part of this ranking process can be performed off line, before the query is submitted through

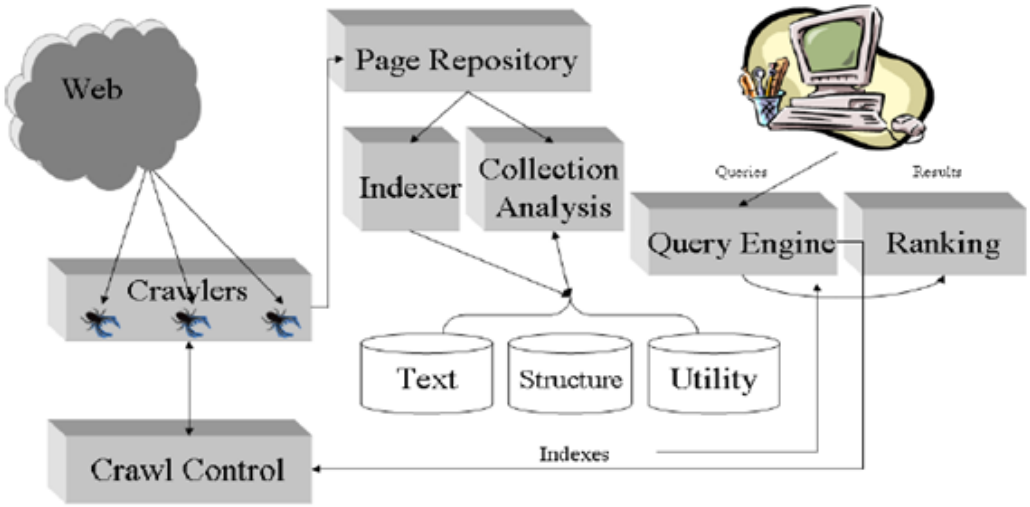

Fig. 1. Architecture of a Search Engine

the query engine. Nowadays, modern search engines index billions of objects on distributed platforms of thousands of commodity PCs running Linux. The index is often organized in a two tier structure and is replicated by thousands of autonomous clusters of servers. This distribution is needed to sustain the peak load of thousands of queries per second.

Users communicate with the query processor, which is the only visible component. It carries out several tasks, usually (but not limited to):

- tokenizing of the query to remove invalid characters, and to recognize meta-keywords or special syntactic operators.
- removal of stopwords; words which are too common and rarely help in the search (e.g. the, a, of, to, which).
- stemming; a process designed to improve the performance of IR systems, involving normalizing semantically similar words to their root forms (e.g. produce, produced, producer, producers, produces and producing map to produc-).
- assigning a weight to each keyword/keyphrase, to aid with ranking(Salton & Buckley 1988).

After results are retrieved by the matching function, they are ranked by relevance based on some ranking measure and set of heuristics (called the ranking scheme). Often taken into account are:

- term frequency how many times keywords appear in the document.
- inverted document frequency a value which aims to determine how important a term is in discriminating a document from others(Salton 1989).
- semantic proximity words synonymous to a given keyword may be matched, boosting the score of the document.
- term position keywords appearing in the title or heading (rather than the body) should contribute more to a document's weight.
- term proximity a document in which the query terms are close together is considered more relevant than one in which they are far apart.
- cluster distance how far apart groupings of matched terms are.
- percentage of query terms matched

4. Web Clustering Engines

Table 2. Comparison of Web Clustering Engines

Name	Time Complexity	Algorithm	Clustering	Reference
Grouper	O(n)	STC	Flat	Zamir and Etzioni 1999
Carrot	O(n)	Lingo	Flat	Weiss and Stefanowski 2003
Vivisimo	$O(n^2)$		Hierarchical	
WICE	O(n)	SHOC	Hierarchical	Zhang and Dong 2004
Web Cat	O(nkt)	K Means	Flat	F. Giannotti et al. 2003
SnakeT	O(n log n + m log mp)	Approximate Sentence Coverage	Hierarchical	Paolo Ferragina and Antonio Gulli 2005

4.1. Grouper

Grouper is a document clustering interface to the HuskySearch meta-search service. HuskySearch (which is based on MetaCrawler [Selberg (1995)]) retrieves results from several popular Web search engines, and Grouper clusters the results as they arrive using the STC algorithm. In the previous Chapter, we present a brief overview of STC and describe its characteristics. Next, we describe the user interface of the Grouper system. A Grouper session starts with the user entering her query in the query box. The user can choose how query terms are treated (e.g., as a phrase, etc.), and can specify the number of documents to be retrieved (10 – 200) from each of the participating search engines. As the system queries approximately 10 search engines, it will typically retrieve 70 – 1000 documents, after eliminating duplicates. After all search engines have returned (or 10 seconds have passed), the main results page is displayed. The main results page displays the number of documents retrieved and the number of clusters found. The clusters are presented in a large table – each cluster in a single

row referred to as the *summary* of the cluster. The clusters are ordered by their estimated coherence. A summary of a cluster includes its size (the number of documents it contains), and attempts to convey the content of the documents in the clusters by displaying *shared phrases* and *sample titles*. Shared phrases are phrases that appear in many documents of the cluster. The numbers appearing in parenthesis after each phrase indicate the percentage of the documents in the cluster that contain the phrase. Titles of three sample documents are also displayed in the summary of each cluster.

4.2. Carrot2

Carrot2 [Stefanowski, J. and Weiss, D (2003)] combines several search results clustering algorithms: STC, Lingo, TRSC, clustering based on swarm intelligence (ant-colonies), and simple agglomerative Techniques. Lingo uses SVD as the primary mechanism for cluster label induction. Carrot2 often tends to create a number of folders which exceeds the number of snippets, thus impacting negatively onto the usability of such software. Carrot2 also fails to cluster together similar labels such as “knowledge, knowledge discovery”, “mining and knowledge”, and furthermore it labels the hierarchy paths with sentences which are one the substring of the other thus introducing few additional knowledge during the browsing.

4.3. Vivisimo

The first commercial clustering engine was probably Northern Light, at the end of the 1990s. It was based on a predefined set of categories, to which the search results were assigned. A major breakthrough was then made by Vivísimo, whose clusters and cluster labels were dynamically generated from the search results. Vivísimo was founded by research computer scientists at the Computer Science Department at Carnegie Mellon University, where research was originally done under grants from the National Science Foundation. The company was founded in June 2000.It won the “best meta-search engine award” assigned by SearchEngineWatch.com from 2001 to 2003. It uses a specially developed heuristic algorithm to group - or cluster - textual documents. This algorithm is based on an old artificial intelligence idea: a good cluster - or document grouping - is one, which possesses a good, readable description. Their document clustering and meta-search software automatically categorizes search results on-the-fly into hierarchical clusters. Vivisimo Velocity is built on a modern architecture and takes advantage of XML and XSL standards. Configuration can also be done through an extensible set of REST/SOAP APIs. In addition, Vivisimo’s solution supports the ability to handle terabytes of information with minimal infrastructure costs compared to other solutions on the market. Through web services and SOA protocols, Vivisimo’s solutions can easily pull data from existing applications, creating a universal gateway to securely access information among disparate systems within an organization.

4.4. WICE (SHOC)

WICE (Web information clustering engine) devise an algorithm called SHOC (semantic hierarchical online clustering) that handles data locality and successfully deals with large alphabets. For solving the first problem, SHOC [Dong 2002; Zhang and Dong 2004] uses suffix arrays instead of suffix trees for extracting frequent phrases.

4.5. WebCAT

WebCAT [Giannotti et al. 2003], was built around an algorithm for clustering categorical data called *transactional k-Means*. Originally developed for databases, this algorithm has little to do with transactional processing and is rather about careful definition of (dis)similarity (Jaccard coefficient) between objects (documents) described by categorical features (words). WebCAT’s computational complexity is linear in the number of documents to be clustered, assuming a fixed number of iterations.

4.6. SnakeT

SnakeT [Ferragina and Gulli 2004, 2005] is both the name of the system and the underlying algorithm. An additional interesting feature of SnakeT is that it builds a hierarchy of possibly overlapping folders. It introduced novel features called approximate sentences—in essence non continuous phrases (phrases with possible gaps). SnakeT’s authors took their algorithm one step further by expanding the potential set of cluster labels with phrases acquired from a predefined index. The computational complexity of the algorithm is $O(n\log_2$

$n + m\log_2 mp$), where n is the number of documents, m is the number of features, and p is the number of labels. Table 2. Compares some of the clustering engines.

5. Comparison of Document Clustering and Web Clustering Engines

Clustering algorithms are developed in order to cluster the documents both in database management system and web page organization in the Search Engines. Several clustering engines are developed by these algorithms and it clusters the web pages retrieved by the search engines automatically. Some are functioning as metasearch engine like metacrawler.com. Others are like Grouper and Retriever etc are the research test bed of the clustering techniques.

Table 3. Document Clustering Vs Web Clustering Engine

Clustering Type	Input	Online	Cluster Label	GUI	Overlap
Document Clustering	Documents	No	Centroid	No	Crisp Clusters
Web Clustering Engine	Snippets	Yes	Natural Language	Yes	Fuzzy Clusters

Document clustering was proposed to improve the effectiveness of document ranking following the hypothesis that closely associated documents will match the same requests [van Rijsbergen 1979]. This approach is used to cluster the entire collection in advance, typically into a hierarchical tree structure, and then return the documents contained in those clusters that best match the user's query based on the scores obtained.

Alternatively, the web clustering engines group the ranked results and gives the user the ability to choose the groups of interest in an interactive manner [HEARST 1996]. It usually refers to browsing a clustered collection of search results returned by a conventional Web search engine. The input and output of a web clustering engine 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, i.e., a URL, a title, and a snippet (a short text summarizing the context in which the query words appear in the result page). The output is a set of labeled clusters representing it in the closest possible way and organized in a set of flat partitions, hierarchy or other graph structure. The main differences between search results clustering and traditional document clustering are summarized in Table 3.

6. Performance Evaluation

Lancaster and Fayen (1973) once listed 6 criteria for assessing the performance of information retrieval systems. They are: 1) Coverage, 2) Recall, 3) Precision, 4) Response time, 5) User effort, and 6) Form of output. Although the criteria were set up more than two decades ago and a great deal has been done to reduce user effort (e.g., design friendly user interface) in using the system, they still seem quite applicable to evaluating information retrieval systems today.

Based on our knowledge and experience gained from the current study, we believe that one needs to consider the following aspects when evaluating a Web search engine.

- Composition of Web Indexes

Whenever a Web search request is issued, it is the web index generated by Web robots or spiders, not the web pages themselves, that has been used for retrieving information. Therefore, the composition of Web indexes affects the performance of a Web search engine. There are three components that the authors would like to inspect regarding the makeup of a Web index, namely, coverage, update frequency and the portions of Web pages indexed (e.g., titles plus the first several lines, or the entire Web page). We understand that the magnitude of all three components depends largely on the power and sophistication of the hardware and software that make the Web index or database. On the other hand, larger coverage, frequent updates and fulltext indexing do not necessarily mean better Web search engines in other measurements.

- Search Capability

A competent Web search engine must include the fundamental search facilities that Internet users are familiar with, which include Boolean logic, phrase searching, truncation, and limiting facilities (e.g., limit by field).

- Retrieval Performance

Retrieval performance is traditionally evaluated on three parameters: precision, recall and response time. While the three variables can all be quantitatively measured, extra caution should be exercised when one judges the relevance of retrieved items and estimates the total number of documents relevant to a specific topic in the Web system.

- Output Option

This evaluation component should be examined from two perspectives. One is the number of output options a Web search engine offers, whereas the other deals with the actual content of the output. Sometimes, one search engine may appear quite impressive in one aspect, but in reality it cannot satisfy its users because of its weakness in the other facet of this evaluation criterion.

- User Effort

User effort refers to documentation and interface in this study. Well-prepared documentation and a user-friendly interface play a notable role in users' selection of Web search engines. Since there are more than two dozen of them available, the attractiveness of each Web search engine is expressed, to its users, mainly in its documentation and interface.

7. Conclusion

We have discussed the issues that must be addressed to build a Web clustering engine and have reviewed a number of existing algorithms, systems and performance measures of search engines. Although there is already much research conducted on the field of web document clustering, it is clear that there are still some open issues that call for more research. To improve the search result clustering, First, more work needs to be done to improve the quality of the cluster labels and the coherence of the cluster structure. Second, the incrementality, because the web pages change very frequently and because new pages are always added to the web. Third, the fact that very often a web page relates to more than one subject should also be considered and lead to algorithms that allow for overlapping clusters. Fourth, Inconsistency is another problem. The contents of a cluster do not always correspond to the label and the navigation through the cluster sub hierarchies does not necessarily lead to more specific results. Fifth, advanced visualization techniques might be used to provide better overviews and guide the interaction with clustered results.

References

[1] Broder, A. (2002) A taxonomy of web search. SIGIR Forum. 36:2. p. 3-10
[2] Cutting, D.R., Karger, D.R., Pedersen, J.O., Tukey, J.W. 1992. Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections. Proc. ACM SIGIR Conference on Research and Development in Information Retrieval, pp.318-329
[3] E. Selberg and O. Etzioni, Multi-service search and comparison using the MetaCrawler, in: Proceedings of the 4th World Wide Web Conference (WWW4), 1995.
[4] FERRAGINA, P. AND GULLI, A. 2004. The Anatomy of SnakeT: A Hierarchical Clustering Engine for Web Page Snippets. In Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases. Lecture Notes in Computer Science, vol. 3202. Springer, 506–508.
[5] GIANNOTTI, F., NANNI, M., PEDRESCHI, D., AND SAMARITANI, F. 2003. WebCat: Automatic categorization of Web search results. In Proceedings of the 11th Italian Symposium on Advanced Database Systems (SEBD), S. Flesca, S. Greco, D. Sacc` a, and E. Zumpano, Eds. Rubettino Editore, 507–518.
[6] HEARST, M. A. AND PEDERSEN, J. O. 1996. Reexamining the cluster hypothesis: scatter/gather on retrieval results. In Proceedings of the 19th ACM International Conference on Research and Development in Information Retrieval. ACM Press, 76–84.
[7] Lancaster, F.W., and Fayen, E.G. (1973). Information Retrieval On-Line Los Angeles, CA: Melville Publishing Co. Chapter 6.
[8] Processing and Management 24(5), 513– 523.
[9] Rasmussen, E. 1992. Clustering Algorithms. Information Retrieval, W.B. Frakes & R. Baeza-Yates, Prentice Hall PTR, New Jersey
[10] Salton, G. & Buckley, C. (1988), 'Term-weighting approaches in automatic text retrieval', Information
[11] Salton, G. (1989), Automatic Text Processing: the Transformation, Analysis, and Retrieval of Information by Computer, Addison-Wesley Longman Publishing Co., Inc.

[12] STEFANOWSKI, J. AND WEISS, D. 2003a. Carrot2 and language properties in Web search results clustering. In Proceedings of the 1st International Atlantic Web Intelligence Conference. Lecture Notes in Computer Science, vol. 2663. Springer, 240–249.
[13] Steinbach, M., G. Karypis, G., Kumar, V. 2000. A Comparison of Document Clustering Techniques. KDD Workshop on Text Mining
[14] Van Rijsbergen, C. J.. 1979. Information Retrieval. Butterworths
[15] Vivisimo.com. http://www.vivisimo.com
[16] Voorhees, E. M. 1986. Implementing agglomerative hierarchic clustering algorithms for use in document retrieval. Information Processing & Management, 22:465-476
[17] Willett, P. 1988. Recent Trends in Hierarchic document Clustering: a critical review. Information & Management, 24(5):577-597
[18] ZAMIR, O. AND ETZIONI, O. 1999. Grouper: A dynamic clustering interface to Web search results. Comput. Netw. 31, 11–16, 1361–1374.
[19] Zamir, O., Etzioni, O. 1998. Web document clustering: a feasibility demonstration. Proc. of SIGIR '98, Melbourne, Appendix-Questionnaire, pp.46-54
[20] ZHANG,D. ANDDONG,Y. 2004. Semantic, hierarchical, online clustering of Web search results. In Proceedings of 6th Asia-PacificWeb Conference (APWeb). Lecture Notes in Computer Science, vol. 3007. Springer, 69– 78.