International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 04 Issue: 08 | Aug -2017 www.irjet.net p-ISSN: 2395-0072

# PERCEPTION DETERMINED CONSTRUCTING ALGORITHM FOR DOCUMENT CLUSTERING

**[*1]Ms.Mageswari M., [*2] Mrs. Shoba S.A.**

*[*1]M.Phil Research Scholar, PG & Research Department of Computer Science & Information Technology Arcot Sri MahalaskshmiWomen's College , Villapakkam, Tamil Nadu, India*
*[*2]. Head of the Department, PG & Research Department of Computer Science & Information Technology Arcot Sri Mahalaskshmi Women's College, Villapakkam, Tamil Nadu, India*

---

**Abstract:** *In the age of increasing information availability, many techniques, such as document clustering, Web search result clustering and information visualization, have been developed to ease understanding of information for users. Organizing Web search results into clusters ease users' quick browsing through search results. However, most of Traditional clustering techniques do not help users directly under-stand key concepts and their semantic relationships in document corpora, which are critical for capturing their conceptual structures. These clustering techniques are least adequate as they don't suggest clusters with highly readable names. Therefore, we present a novel approach called 'Se-mantic Lingo' to identify the key concepts and automatically generate ontology based on these concepts for conceptualization of document.*

**Keywords: Concept Driven Semantic, Document Clustering, Cluster Merging, K-means clustering.**

## I. INTRODUCTION

The amount of available information on the Web is increasing rapidly. The publicly index able Web contains an estimated 800 million pages as of February 1999, encompassing about 15 terabytes of information or about 6 terabytes of text after removing HTML tags, comments, and extra whitespace As noted by Lawrence and Giles, "the revolution that the Web has brought to information access is not so much due to the availability of information (huge amounts of information has long been available in libraries and elsewhere), but rather the increased efficiency of accessing information, which can make previously impractical tasks practical"[3].

As of December 1998, 85% of Web users used search services to locate Web pages and 60% used Web directories On the other hand, in the same survey 45% of the users stated that one of the biggest problems of using the Web was the inability to find the information they were looking for. There are some well-studied reasons why finding information using Web search engines is not always successful[7]. No single search engine has indexed the entire Web. As of August 1999, FAST reports to have indexed the largest number of pages - 200 million (less than 25% of the estimated size of the index able Web), while an independent study showed no search engine to index more than 16% of the Web. Another factor that decreases search engine usefulness is the dynamic nature of the Web, resulting in many "dead links" and "out of date" pages that have changed since indexed. But even accepting these factors, finding relevant information using Web search engines often fails.

Document retrieval systems typically present search results in a ranked list, ordered by their estimated relevance to the query. The relevancy is estimated based on the similarity between the text of a document and the query. Such ranking schemes work well when users can formulate a well-defined query for their searches. However, users of Web search engines often formulate very short queries (70% are single word queries) that often retrieve large numbers of documents. Based on such a condensed representation of the users' search interests, it is impossible for the search engine to identify the specific documents that are of interest to the users[2][11]. Moreover, many webmasters now actively work to influence rankings. These problems are exacerbated when the users are unfamiliar with the topic they are querying about, when they are novices at performing searches, or when the search engine's database contains a large number of documents. All these conditions commonly exist for Web search engine users. Therefore the vast majority of the retrieved documents are often of no interest to the user; such searches are termed low precision searches.

## II. EFFECTIVE WORK

Document clustering has been recognized as a central problem in text data management, and it becomes particularly challenging when documents have multiple topics. In this paper we address the problem of multi-topic document clustering by leveraging the natural composition of documents in text segments, which bear one or more topics on their own [4]. We propose a segment-based document clustering framework, which is designed to induce a classification of documents starting from the identification of cohesive groups of segment-

based portions of the original documents. We empirically give evidence of the significance of our approach on different, large collections of multi-topic documents.Andrea Tagarelli, George Karypis [1]. The data structure used to describe rough clusters. It also provides an overview of the evolutionary algorithm used to develop viable cluster solutions, consisting of an optimal number of templates providing descriptions of the clusters. This algorithm was tested on a small data set and a large data set.Kevin E. Voges [2].

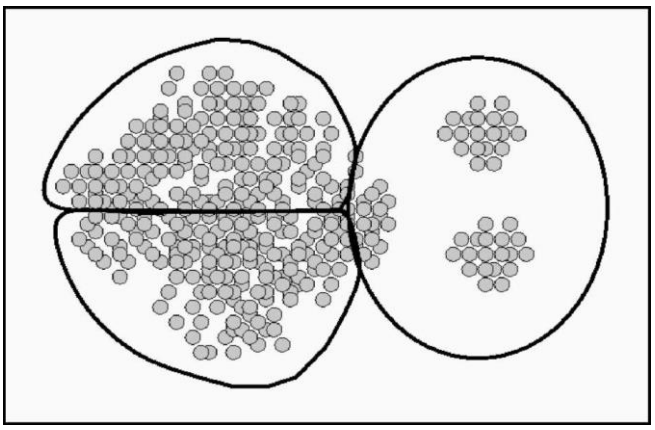

**Fig: Clustering data from unequal-sized Clusters**

Hierarchical clustering is often portrayed as the better quality clustering approach, but is limited because of its quadratic time complexity. In contrast, K-means and its variants have a time complexity which is linear in the number of documents, but are thought to produce inferior clusters. Sometimes K-means and agglomerative hierarchical approaches are combined so as to "get the best of both worlds [11]." However, our results indicate that the bisecting K-means technique is better than the standard K-means approach and as good as or better than the hierarchical approaches that we tested for a variety of cluster evaluation metrics.Michael Steinbach George KarypisVipin Kumar [3]. The proposed approach exploits the way we can identify the theme of the document based on disambiguated core semantic features extracted and exploits the characteristics of lexical chain based on WordNet. In our approach the main contributions are preprocessing of document which identifies the noun as a feature by performing tagging and lemmatization, performing word sense disambiguation to obtained candidate words based on the modified similarity approach and finally the generation of cluster based on lexical chains. Shabanaafreen, Dr. B. Srinivasu [4]

## III. PREVIOUS IMPLEMENTATIONS

Extensive growth in information technology and its exponential use has generated myriad of data in the forms of text documents. In order to have fast retrieval from such extensive data and to facilitate effective browsing and searching it is vital to organize documents. Search engines are considered as the most common tool to retrieve information from the Internet. But the search results returned by search engines usually come with huge quantity and a long list. What's more, the search results will be very diverse when users' search term is not correct or ambiguous. It's time-consuming and arduous to find the most relevant search result. Search results clustering are an efficient method to make the search results easier to scan. Search results clustering works on snippets (a summary of the search results), which is different from document clustering (working on long text).Traditional clustering algorithms do not consider the semantic relationships among the words so that can not accurately represent the meaning of documents [2]. To over-come this problem semantic information from ontology such as Domain Ontology has been used to improve the Quality of Web search clustering. Our key goal is to improve our system by overcome various problems such as Synonym and polysemy, high dimensionality and assigning appropriate description for generated cluster.

### 3.1 Increasing Precision of Web Search Results

Document retrieval systems typically present search results ordered by their estimated relevance to the query, which is calculated using IR metrics that capture the similarity between the text of a document and the query. Such ranking schemes work well when users can formulate well-defined queries for their searches. However, users of Web search engines often formulate very short queries (70% are single word queries) that often retrieve a large set of documents. Based on such a condensed representation of the users' search interests, it is nearly impossible to locate within these large document sets the specific documents that are of interest to the user. Moreover, many webmasters now actively work to influence rankings. These problems are exacerbated when the users are unfamiliar with the topic they are querying about, when they are novices at performing IR searches, and when the database contains a large number of documents. All these conditions commonly exist for Web search engine users. To address these problems, recent research has focused on ranking retrieved results based on information other than the text appearing in the documents.

$$PR(p) = (1-d) + d( PR(l1)/C(l1) + ... + PR(ln)/C(ln) )$$

First, Web documents, and especially search engine snippets, are short and often poorly formatted, thus they are difficult to cluster. Second, the algorithms that were sufficiently fast were all model-based algorithms. Model-based algorithms have a priori assumptions as to the model describing the data (e.g., the k-means algorithm assumes spherical and equal-sized clusters). These algorithms search for the most probable model

parameters, given the data and the a priori assumptions regarding the model. When the data does not fit the model these algorithms perform poorly. There is no reason to believe that Web documents fit these models.

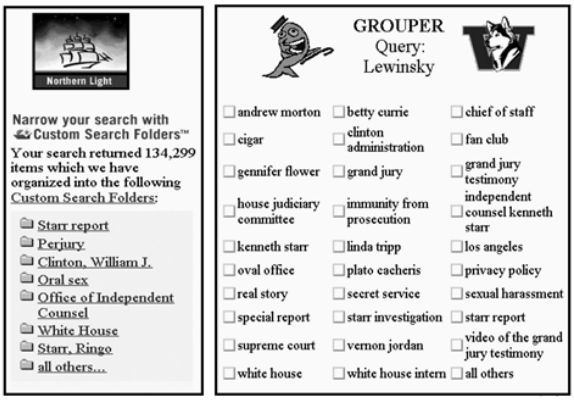


**Fig. 3.1 Helping the User in Low Precision Searches**

Many Web queries result in a very large number of returned documents. Typically, the vast majority of these are of no interest to the user and therefore such searches can be termed *low precision searches.* Most Web search engines provide tools aimed at helping users "cope" with (and make use of) these large document sets.

Many Search engines allow users to sort the results by site (e.g., Excite, Info seek, Hotpot and Lycos), or by date (e.g., InfoSeek and Northern Light). Some search engines allow users to narrow down the set of matches already generated (the "Search Within" feature of Infoseek and Lycos).

### 3.2 User Interfaces to Search Results

Visualization of search results has been investigated as a tool for presenting retrieved documents to the user in ways that can scale to large document sets and provide more information to the user than the ranked list interface. Various visualization techniques were designed to help users get a better comprehension of the returned document set, identify interesting documents faster, and reformulate the query more accurately. Document visualization techniques fall into two broad categories: Visualization of document attributes and visualization of inter document similarities [4]. The visualization of document attributes is designed to display additional information about the retrieved documents. This will often have the secondary effect of grouping documents that share similar attributes.

## IV. PROPOSED ANALYSIS

First, a document corpus is preprocessed into term frequency files, in which each document is represented as a list of its term frequencies. In addition, common phrases are extracted us-ing a suffix array algorithm . Second, the inverted document frequency of each term is calculated and each term weight is computed by multiplying the term frequency and inverted document frequency. Inverted term-document files are generated for each term and the term-document matrix is constructed based on term weights. Third, with the extracted common phrases, we conduct key concept induction using LSA techniques. Fourth, with the list of key concepts, we utilize WordNet to inspect their synonyms and hyponyms. The documents are allocated based on each key concept and its synonyms and hyponyms. Fifth, using WordNet, hypernyms of each concept are detected and used to construct a corpus-related ontology. Sixth, documents are linked to the ontology through the key concepts.

we use the vector space model (VSM) and singular value decomposition (SVD), the latter being the fundamental mathematical construct underlying the latent semantic analysis (LSA) technique. VSM is a method of information retrieval that uses linear-algebra operations to compare textual data, associating a single multidimensional vector with each document in a collection, and each component of that vector reflects a particular keyword or term related to the document. LSA aims to represent the input collection using abstract terms found in the documents rather than the literal terms appear-ing in them, by approximating the original term-document matrix using a limited number of orthogonal factors. These factors represent a set of abstract terms, each conveying some idea common to a subset of the document corpora.

### 4.1 Construction Algorithm

- Let Ni denote the intermediate tree that encodes all the suffixes from 1 to i.
- Tree N1 consists of a single edge between the root of the tree and a leaf labeled 1. The edge is labeled with the string S.
- Tree Ni+1 is constructed from Ni as follows:
- Starting at the root of Ni the algorithm finds the longest path from the root whose label matches a prefix of S[i+1..m]. This path is found by successively comparing and matching words in suffix S[i+1..m] to words along a unique path from the root, until no further matches are possible.
- When no further matches are possible, the algorithm is either at a node, say w, or it is in a middle of an edge.
- If it is in the middle of an edge, say (u,v), then it breaks (u,v) into two edges by inserting a new node w just after the last word on the edge whose label matches a prefix of the suffix S[i+1..m].
- In either case, the algorithm creates a new edge (w,i+1) running from w to a new leaf labeled i+1,

and labels the new edge with the unmatched part of suffix S[i+1..m].

To sum, suffix trees can be constructed in linear time, and furthermore, Ukkonen's algorithm allows this to be done online, as the strings are read into memory. One caveat has to be mentioned: for large trees, paging can be a serious problem because the trees do not have nice locality properties. Indeed, by design, suffix links allow the algorithm to move quickly from one part of the tree to a distant part of the tree. This is great for worse case time analysis but is horrible for paging if the tree does not fit entirely in memory. Consequently, great efforts are often spent to reduce the space requirements of suffix trees. With this in mind, a new data structure, called a suffix array, was proposed . Suffix arrays are very space efficient and can be used to solve many string-matching problems almost as efficiently as suffix trees

### 4.2 Phrase Cluster Merging

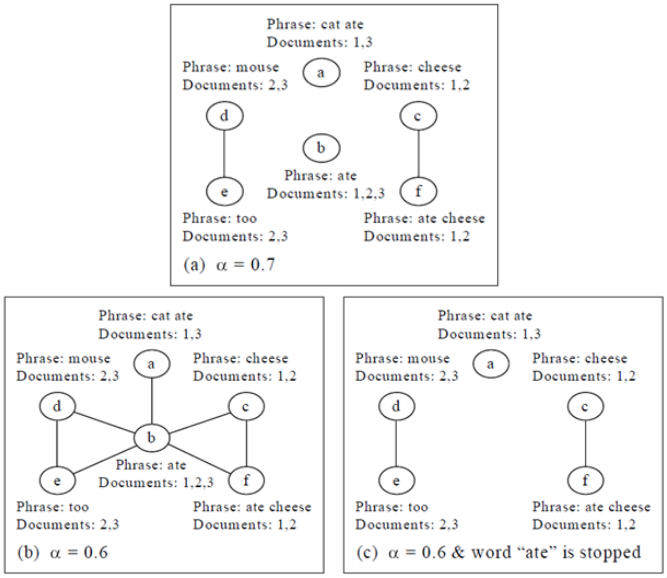


**Fig.4.1 The phrase cluster graph**

In the previous step, we have identified groups of documents that share phrases. However, documents may share more than one phrase. Therefore, the document sets of distinct phrase clusters may overlap and may even be identical. To avoid the proliferation of nearly identical clusters, the third step of the STC algorithm merges phrase clusters with a high overlap in their document sets.

The generalized suffix tree of the three strings "cat ate cheese", "mouse ate cheese too" and "cat ate mouse too". The internal nodes of the suffix tree are drawn as circles, and are labeled a through f for further reference. There are 11 leaves in this example (the sum of the lengths of all the strings) drawn as rectangles. The first number in each rectangle indicates the string from which that suffix originated; the second number represents the position in that string where the suffix starts.

### 4.3 Clustering Methods

#### K-Means

K-means is the most important flat clustering algorithm. The objective function of K-means is to minimize the average squared distance of objects from their cluster centers, where a cluster center is defined as the mean or centroid j of the objects in a cluster C:

The ideal cluster in K-means is a sphere with the centroid as its center of gravity. Ideally, the clusters should not overlap. A measure of how well the centroids represent the members of their clusters is the Residual Sum of Squares (RSS), the squared distance of each vector from its centroid summed over all vectors

$$RSSi = \; Z|| X\text{-}M(C)||2 \; RSS = Z \; RSS,$$

K-means can start with selecting as initial clusters centers K randomly chosen objects, namely the seeds. It then moves the cluster centers around in space in order to minimize RSS. This is done iteratively by repeating two steps until a stopping criterion is met

#### Algorithm for K-Means

**Procedure** KMEANS(X,K)
{s1, s2, • • • ,sk} SelectRandomSeeds(K,X)
**for**i <-1,K **do**
u(Ci) ^ si
**end for**
**repeat**
**min**k~xn|j(Ck)k **C**k**= C**k**[ {~X**n}
**for all C**k**do**
**j(C**k**) = 1**
**end for**
**until** stopping criterion is met
**end procedure**

### 4.4 Hierarchical Clustering

Hierarchical clustering approaches attempt to create a hierarchical decomposition of the given document collection thus achieving a hierarchical structure. Hierarchical methods are usually classified into Agglomerative and Divisive methods depending on how the hierarchy is constructed.

- Document Parsing, in which each document is transformed into a sequence of words and phrase boundaries are identified.

- Phrase Cluster Identification, in which a suffix tree is used to find all maximal phrase clusters. These phrase clusters are scored and the highest scoring ones are selected for further consideration.
- Phrase Cluster Merging, in which the selected phrase clusters are merged into clusters, based on the overlap of their document sets.

## V. EVALUATION RESULT

The performance of the STC algorithm and comparing it to other commonly used (vector-based) clustering algorithms. We start by describe the three document collections used in our experiments: two collections of Web Search Results and the OHSUMED medical abstracts collection. In section 4.3 we evaluate STC's "clustering quality". This is done using two quality-evaluation approaches - the IR approach and the merge-then-cluster approach. In the following section we investigate the importance of overlapping clusters and of the use of phrases for STC by evaluating hobbled versions of the STC algorithm without these features. We also investigate whether overlapping clusters can improve the performance of other clustering algorithm.

| *Node* | *Phrase* | *Documents* |
|---|---|---|
| a | cat ate | 1,3 |
| b | ate | 1,2,3 |
| c | cheese | 1,2 |
| d | mouse | 2,3 |
| e | too | 2,3 |
| f | ate cheese | 1,2 |

**Table1 : The six internal nodes from the example shown in Figure 3.2 and their corresponding phrase clusters**

After creating the suffix tree, we determine which nodes correspond to the maximal phrase clusters, and assign to each of these a score that is a function of the number of documents it contains, and its phrase. This is done in a single traversal of the tree. The score of a phrase cluster attempts to estimate its usefulness for our clustering task. The score s(m) of maximal phrase cluster m with phrase mp is given by

After evaluating the STC algorithm we explore the question: Will phrases also improve the clustering quality of other clustering algorithms besides STC, and if so why? This question will be addressed in section 4.5. After showing that phrases greatly improve performance across all algorithms we investigate where the advantage of using phrases comes from. Is it simply their being multiword features or is the adjacency and order information also useful? To address this we investigate a variation of STC thatuses frequent sets as multiword features instead of phrases. We also compare n-grams to suffix tree phrases to evaluate the added importance of using long phrases for clustering.

$$s(m) = |m| \cdot f(|mp|) \cdot 6\ t_{fi}d_f(w_i) \text{ ----(1)}$$

### A) 20 newsgroups

This is a very standard and popular dataset used for evaluation of many text applications, data mining methods, machine learning methods, etc. Its details are as follows:

- Number of unique documents = 18,828
- Number of categories = 20
- Number of unique words after removing the stopwords = 71,830

### B) Reuters -21578

This is the most common dataset used for evaluation of document categorization and clustering. Its details are as follows:

- Number of unique documents = 19715
- Number of categories = 5
- Number of unique words after removing the stopwords = 39,096

It contains many sub-categories also but for this experiment I am using only the broad categories.

### C) Keep media dataset

This is a set of news articles provided by a company. Its details are as follows:

- Number of unique documents = 62,239
- Number of categories = 69
- Number of unique words after removing the stop words = 3,36,656

| Number of clusters | Entropy extracted | Entropy remaining |
|---|---|---|
| 10 | 0.62 | 0.96 |
| 50 | 0.57 | 0.83 |
| 100 | 0.49 | 0.59 |
| 200 | 0.48 | 0.56 |
| 500 | 0.36 | 0.47 |
| 1000 | 0.29 | 0.36 |

**Table 2: Similarity threshold = 0.5 Number of extracted documents = 23,456**

| Number of Clusters | Entropy with use of Graclus | Entropy with use of Metis |
|---|---|---|
| 100 | 2.48 | 2.42 |
| 500 | 1.79 | 1.70 |
| 1000 | 1.30 | 1.32 |

**Table 3 : Reuters - 21578 dataset**

### 5.1 Clustering Quality

Here evaluate the "clustering quality" of STC as compared to other clustering algorithms and to the ranked list presentation of the search results. The clustering algorithms compared are Buckshot, Fractionation, group-average hierarchical clustering (GAVG), k-means, single-pass and STC. The GAVG algorithm was chosen as it is commonly used; the rest were chosen as they are fast enough to be contenders for online clustering. The clustering quality is compared using two quality-evaluation approaches: the IR approach (for all three collections) and the "merge-then-cluster" approach (for the OHSUMED collection).

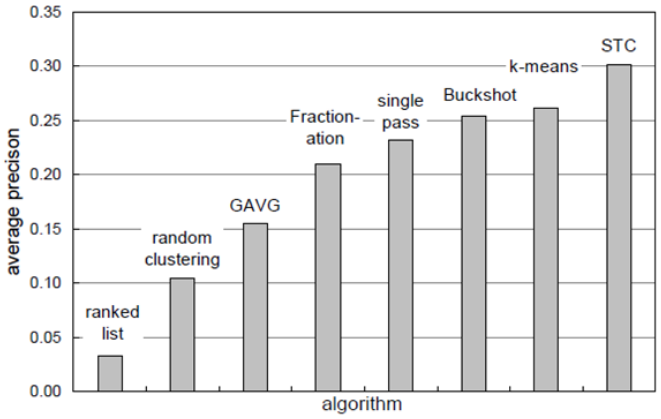


**Fig.5.1 Average precision on the WSR-SNIP collection**

Compare the average precision for the results of the six clustering algorithms, the original list and random clustering, given the 10%-viewed user model. Compare the algorithms on the WSR-SNIP, WSR-DOCS and OHSUMED collections respectively. As can be seen from the results, STC outperform all other algorithms, except for the k-means algorithm on the OHSUMED collection. Note that the performance of the algorithms varies greatly across the different document collections.

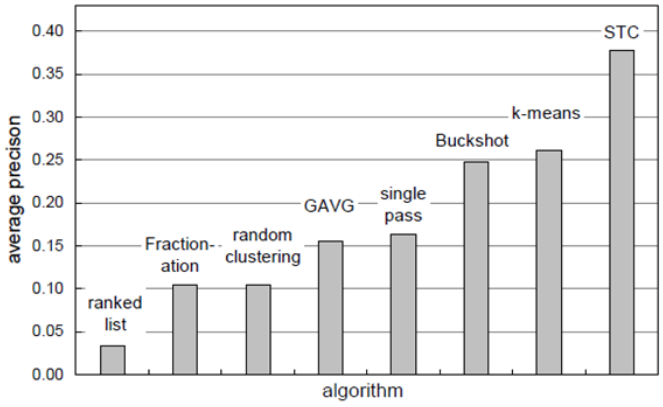


**Fig.5.2 Average precision on the WSR-DOCS collection**

The average precision of the six clustering algorithms, the original list and random clustering, given the 10%-viewed user model, on the WSR-DOCS collectionStatistical significance tests were performed using the two-tailed paired sign test. We have only 10 document sets in the WSR collections and therefore for two algorithms to be statistically different from each otherone has to outperform the other in 9 out of the 10 run. On the WSR-SNIP collection, the difference between STC and single-pass was significant, but the difference to k-means and Buckshot was not. On the WSR-DOCS collection, STC did outperform all other algorithms with statistical significance.

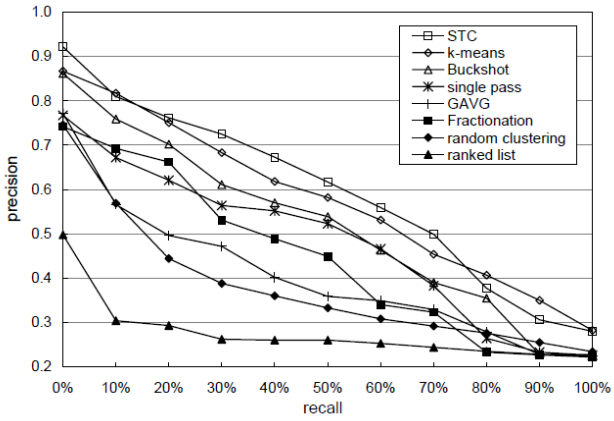


**Fig5.3: Precision-Recall Graph for the WSR–SNIP Collection**

Present the precision-recall graphs of the different algorithms for the three text collections. The "STC-min" and the "STC-max" evaluation methods were averaged into a single STC measurement in the interest of clarity, as the difference between them was always quite small (~5%). The three random clustering's (with different size distributions) exhibited practically identical precision-recall graphs and therefore only one is shown.

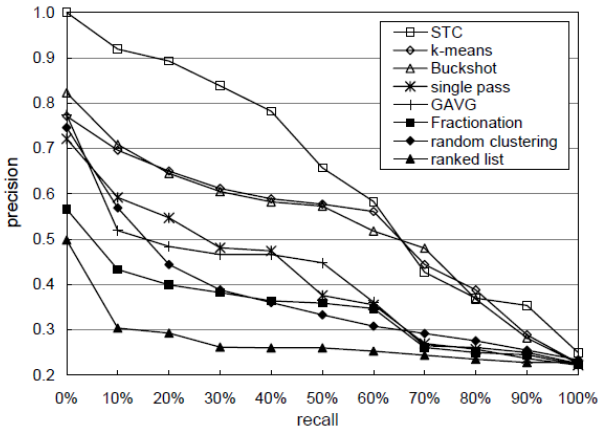


**Fig5.4 : Precision-recall graph for the WSR–DOCS collection**

It present the Number-to-View graphs of the six algorithms, the ranked list and the random clustering. The Number-to-View graphs plot the Expected Search Length (the expected number of irrelevant documents that have to be viewed in order to encounter a certain number of relevant documents) as a function of the number of relevant documents the user wishes to view (1-10).

## CONCLUSION

The amount of available information on the Web is increasing rapidly and users rely on search engines to find the information they are looking for. However, finding relevant information using Web search engines often fails. One of the reasons for this is that users typically submit queries that are short and general, retrieving a large numbers of documents, the vast majority of which are of no interest to the user.

The low precision of the Web search engines coupled with the ranked list presentation force users to sift through a large number of documents and make it hard for them to find the information they are looking for. As low precision Web searches are inevitable, tools must be provided to help users "cope" with (and make use of) these large document sets. The motivation for this research was to make search engine results easy to browse, allowing the user to find a relevant documents in the result set even if it is very large and mostly irrelevant.

## REFERENCES

1. [Agrawal et al., 93] Agrawal, R., Imielinski, T. and Swami, A. Mining associations between sets of items in massive databases. In Proceedings of the ACM-SIGMOD 1993 International Conference on Management of Data (SIGMOD'93), 207-216, 1993.
2. [Agrawal and Srikant, 94] Agrawal, R. and Srikant, R. Fast algorithms for mining association rules. In Proceedings of the 20th International Conference on Very Large Databases (VLDB'94), 1994.
3. [Allan et al., 96]. Allan, J., Ballesteros, L., Callan, J. P., Croft, W. B. and Lu, Z. Recent experiments with INQUERY. In D. K. Harman (ed.), the Fourth Text Retrieval Conference (TREC-4), NIST Special Publication, 1996.
4. [Allen et al., 93] Allen, R. B., Obry, P. and Littman, M., An interface for navigating clustered document sets returned by queries. In Proceedings of the ACM Conference on Organizational Computing Systems (COOCS'93), 166-171, 1993.
5. [Bayardo, 97] Bayardo, R. J. Brute-force mining of high-confidence classification rules. In Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining (KDD'97), 123-126, 1997.
6. [Bayardo, 98] Bayardo, R. J. Efficiently mining long patterns from databases. In Proceedings of ACM SIGMOD International Conference on Management of Data(SIGMOD'98), 85-93, 1998.
7. [Boyan et al., 96] Boyan, J., Freitag, D. and Joachims, T. Machine architecture for optimizing Web search engines. In Proceedings of the AAAI-96 Workshop on Internet based Information Systems, 1996.
8. [Brin et al., 97] Brin, S., Motwani, R., Ullman, J. and Tsur, S. Dynamic itemset counting and implication rules for market basket data. In Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD'97), 255-264, 1997.
9. [Brin and Page, 98] Brin, S. and Page, L. The anatomy of a large-scale hypertextual Web search engine. In Proceedings of the Seventh International Web Wide World Conference (WWW7), 1998.
10. [Broder et al., 97] Broder, A. Z., Glassman, S. C., Manasse, M. S. and Zweig, G. Syntactic clustering of the Web. In Proceedings of the Sixth International Web WideWorld Conference (WWW6), 1997.