International Journal of Engineering Research and Development
e-ISSN: 2278-067X, p-ISSN: 2278-800X, www.ijerd.com
Volume 10, Issue 3 (March 2014), PP.79-86

A Comparative Analysis on Web Clustering Algorithms

Mahalakshmi[1]
[1]*Assistant Professor, Dept. of Computer Science, Sri Ganesh College of Arts & Science, Salem – 636 014, Tamilnadu, India.*

Abstract: - The aim of the thesis work entitled **"A COMPARATIVE ANALYSIS ON WEB CLUSTERING ALGORITHMS"** is to identify the efficient web document clustering algorithm for searching the information from very large databases available in the web efficiently and effectively.A number of clustering algorithms exists for the retrieval of information from large databases. Those algorithms are implemented in several search engines available today like Yahoo, Google and Info seek etc., The purpose of this thesis work is to compare the different clustering algorithms in the field of data mining. The different clustering algorithms used for the comparative analysis are STC (Suffix Tree Clustering), K-means, Fractionation, AHC (Agglomerative Hierarchical Clustering) and SHOC (Semantic Hierarchical Clustering Algorithm).The comparative analysis of web document clustering algorithms uses two parameters. First parameter time is used to mention the time taken to fulfill the given query by the algorithm. The second parameter is used to mention the number of documents retrieved by the algorithm.The purpose of this comparative analysis is to find the efficient and effectiveness s of the different web document clustering algorithms. The suffix Tree clustering algorithm (STC) found to be the best algorithm for the retrieval of maximum documents with minimum time.

Keywords: - CLARANS, CLARA, STC, AHC, SHOC.

1) INTRODUCTION

The world wide web serves as huge, widely distributed, global information Service center for news, advertisements, consumer information, education, financial management, government and many other information services. The web also contains rich and dynamic collection of hyperlink information, providing rich sources for data mining. However, based on the following observations, the web also poses great challenges for effective resource and knowledge discovery. The web seems to be too huge for effective data warehouse and data mining. The Complexity of web pages is far greater than that of any traditional text document Collection. The web is a highly dynamic information source. The web serves a broad diversity of user communities.

Only a small portion of the information on the web is truly relevant or useful. There are many index based web search engines that search the web, index web pages, and built and store huge keyword-based indices that help locate sets ofweb pages containing certain keywords. With such search engines, an experienced user may be able to quickly locate documents byproviding a set of tightly constrained keywords and phrases. However, current keyword-based search engines suffer from several deficiencies. First, a topic of any breadth may easily contain hundreds of thousands of documents. This can lead to a huge number of document entries returned by the search engine, many of which are only marginally relevant to the topic may not contain keywords defining them. This is referred to as the **polysemy problem.**

There are several key requirements for web document clustering algorithms. As efficient algorithm should satisfy the following properties

i) Relevance: The method ought to produce clusters that group documents relevant to the user's query separately from irrelevant ones.

ii) Browsable-Summaries: The user needs to determine at a glance whether a cluster's contents are of interest. Do not want to replace sifting through ranked lists with sifting through clusters. Therefore the method has to provide concise and accurate descriptions of the clusters.

iii) Overlap: Since documents have multiple topics, it is important to avoid confining each document to only one cluster. That is good clustering algorithm return a document in more than one cluster.

iv) 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 of the web.

v) Speed: A very patient user might sift through 100 document in a ranked list Presentation. Clustering allow the user to browse through at least an order of magnitude more document. 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.

vi) Incremental: To save time, the method should start to process each snippet as soon as it is received over the web.

1) Suffix Tree Clustering

Suffix Tree Clustering algorithm is a novel, incremental, O(n) time algorithm designed to satisfy these requirements, where n denotes the number of documents clustered. STC does not treat a document as a set of words but rather as a string, making use of proximity information between words. STC relies on a suffix tree to efficiently identify sets of documents that share common phrases and uses this information to create clusters and to succinctly summarize their contents for users.

Suffix Tree Clustering (STC) is a linear time clustering algorithm that is based on identifying the phrases that are common to group of documents. A phrase is an ordered sequence of one or more words. A base cluster is defined as set of documents that share a common phrase.

STC has three logical steps.

(1) Document "cleaning"

(2) Identifying base clusters using suffix tree and

(3) Combining these base clusters into clusters.

Step1:- Document "Cleaning"

In this step, using various methods cleans the document. The string of text representing each document is transformed using a **light-stemming algorithm** (deleting word prefixes and suffixes and reducing plural to singular). Sentence boundaries (identified via punctuation and HTML tags) a remarked and non-word tokens (such as number, HTML tags and most punctuation) are stripped. The original document strings are kept, as well as pointers from the beginning of each word in the transformed string to its position in the original string. This enables to identify key phrases in the transformed string, to display the original text for enhanced user readability.

Step 2:- Identifying Base Clusters

The identification of base clusters can be viewed as the creation of an inverted Index of phrases for document collection. This is done efficiently using a data structure called a **suffix tree.** This structure can be constructed in time linear with the size of the Collection, and can be constructed incrementally as the documents are being read. A suffix tree of a string S is a compact tier containing all the suffixes of S. Treat the Documents as strings of words, not characters, thus suffixes contain one or more whole words. In more precise terms.

1. A suffix tree is a rooted, directed tree
2. Each internal node has at least 2 children
3. Each edge is labeled with a non-empty sub-string of S (hence it is a tier).

The label of a node is defined to be the concatenation of the edge-labels on the path from the root to that node.

4. No two edges out of the same node can have edge-labels that begins with the same word (hence it is compact)
5. For each suffix s of S, there exists a suffix-node whose label equals s.

The suffix tree of a collection of strings is a compact tier containing all the Suffixes of all the strings in the collection. Each suffix-node is marked to designate from which string or strings it originated from i.e., the label of that suffix-node is a suffix of that string. Construct the suffix tree of all the sentences of all the documents in the Collection. Consider the following example,

"cat ate cheese"

"mouse ate cheese too"and

"cat ate mouse too"

The nodes of the suffix tree are drawn as circles. Each suffix-node has one or more boxes attached to it designating the string(s) it originated from. The first number in each box designates the string of origin (1-3 in the example, by the order the strings appear above); the second number designates which suffix of that string labels that suffix- node. Several of the nodes in the figure are labeled a through f for further reference.

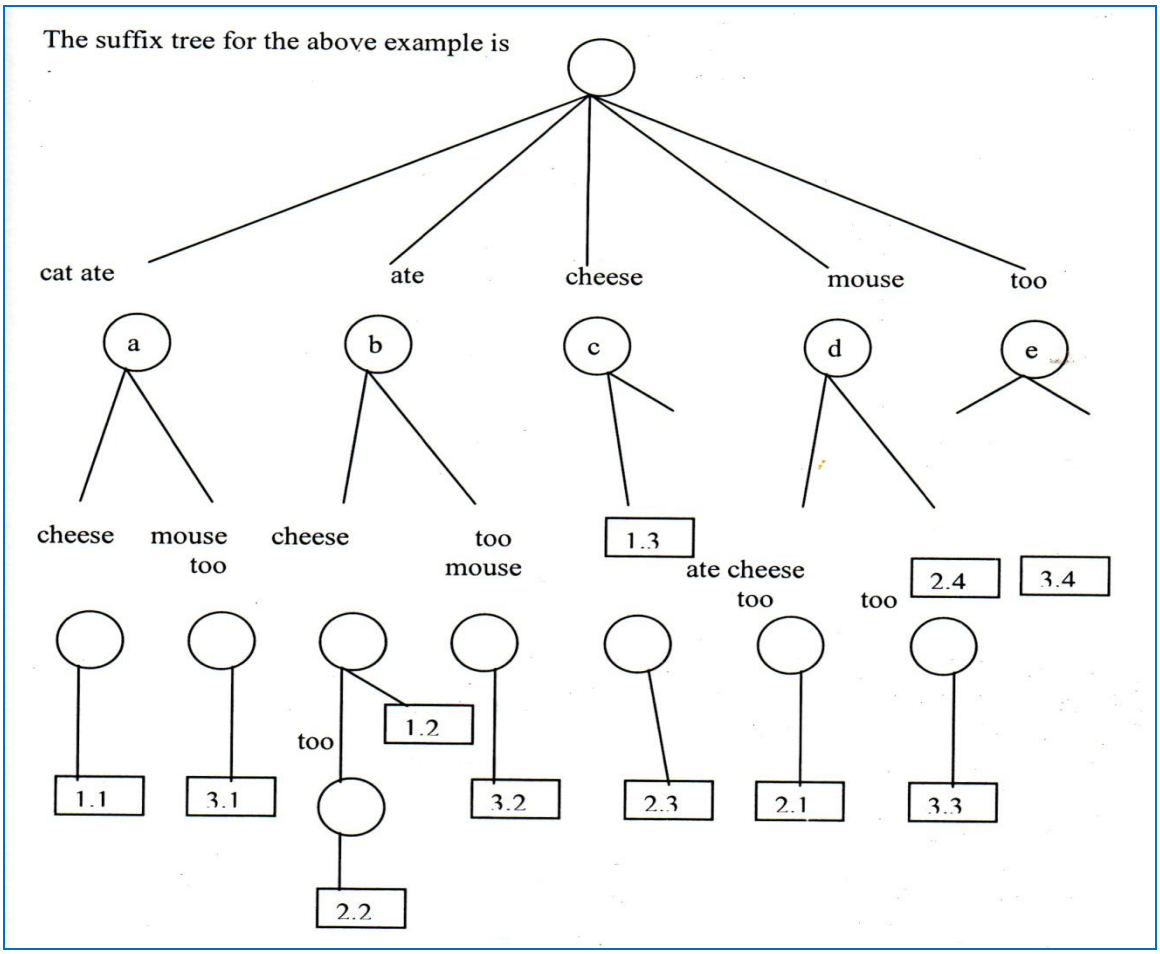


Fig: suffix tree

Each node of the suffix tree represents a group of documents and a phrase that is common to all of them. The label of the node represents the common phrase; the set of documents tagging the suffix-nodes that are descendants of the node make up the document group. Therefore, each node represents a base cluster. Furthermore, all possible base clusters (containing 2 or more documents) appear as nodes in a suffix tree.The following table lists the sox marked nodes from the example shown in the above figure and their corresponding base clusters.

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

Each base cluster is assigned a score that is a function of the number of document contains, and the words that make up its phrase. The score s(B) of base cluster B withPhrase P is given by:S(B)=|B|.f(|P|)
Where |B| is the number of documents in base cluster B,and|P| is the number of words in P that have a non-zero score (i.e., the effective length of the phrase). The stop list is maintained with internet specific words (e.g., "previous", "java", "frames" and "mail"). Words appearing in the stop list , or that appear in too few (3 or less) or too many (more than 40% of the collection) documents receive a score of zero. The function f penalizes single word phrases, is linear for phrases that are two to six words long, and becomes constant for longer phrases.

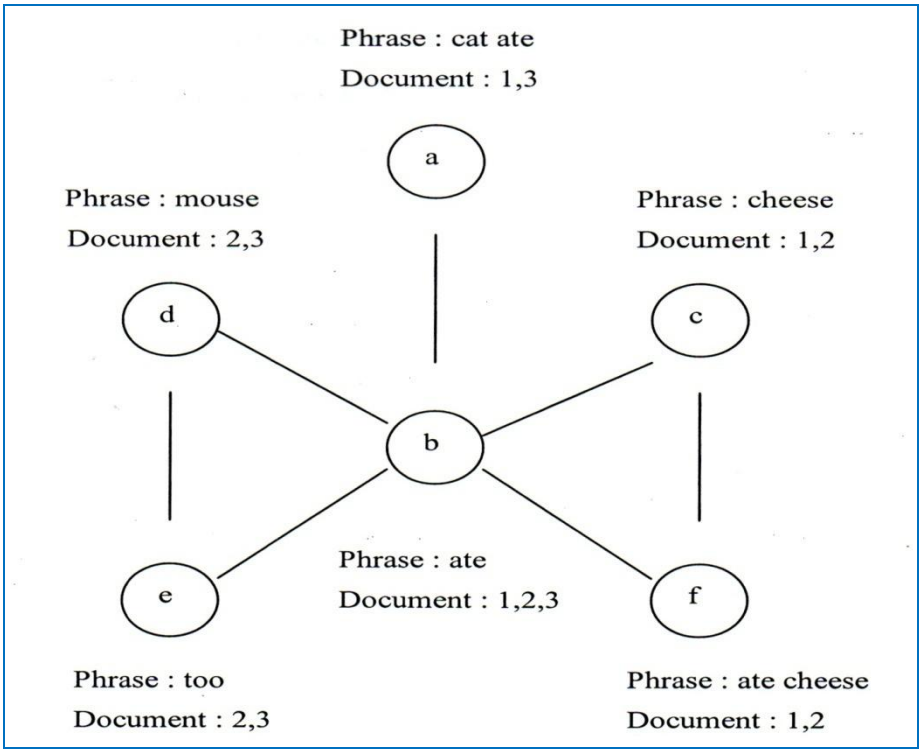
Step 3- combining Base Clusters

Documents may share more than one phrase. As a result, the document sets of distinct are Clusters may overlap and may even be identical. To avoid the **proliferation** of nearly identical clusters, the third step of the

algorithm merges base clusters with a high overlap in their document sets.A binary similarity measure between base-clusters based on the overlap of their document sets. Given two base clusters Bm and Bn, with sizes |Bm| and |Bn| respectively, and |Bm∩Bn| representing the number of documents common to both base clusters, the similarity of
Bm and Bn to be 1 iff:

$|Bm \cap Bn|/|Bm| > 0.5$ and
$|Bm \cap Bn|/|Bn| > 0.5$

Otherwise, their similarity is defined to be 0
Next, in the base cluster graph, where nodes are base clusters, and two nodes are connected if and only if the two base clusters have a similarity of 1. A cluster is defined as being a connected component in the base cluster graph. Each cluster contains the union of the documents of all its base clusters. The following figure illustrates the base cluster graph of the six base clusters in the above table. There is a single cluster in in this example.



In essence, clustering the base clusters using the equivalent of a **single-link clustering algorithm** where a predetermined minimal similarity between base clusters serves as the halting criterion. This clustering algorithm is incremental and order independent. The STC .algorithm is incremental. As each document arrives from the Web, clean it and add it to the suffix tree.

Each node that is updated (or created) as a result of this is tagged. Then update the relevant base clusters and recalculates the similarity of these base clusters to the rest of the base clusters. Finally, check if the changes in the base cluster graph results in any changes to the final clusters. To keep the cost of this last step constant, don't check the similarity of the modified base clusters with all other base clusters, but only with the k highest scoring base clusters (take k to be 500). The cost of "cleaning" the document is obviously linear with the collection size. The cost of inserting documents into the suffix tree is also linear with the collection size, as is the number of nodes that can be affected by these insertions. Thus the overall time complexity of STC is linear with regard to the collection size.The final clusters are scored and sorted based on the scores of their base clusters and their Overlap. As the final number of clusters can vary, report only the top few clusters. Typically, only the top 10 clusters are of interest. For each cluster, report the number of documents it contains, and the phrases of its base clusters.

The goal of a clustering algorithm is to group each document with others sharing a common topic, but not necessarily to partition the collection. It has been claimed that it is artificial to force each document into only one cluster, as documents often have several topics.Such a constraint could decrease the usefulness of the clusters produced. Allowing a Document to appear in more than one cluster acknowledges that documents are complex objects, which may be grouped into multiple potentially overlapping, but internally coherent groups.This is actually the reason many IR system use some form of dot-product document Similarity – measure

it allows a document to be similar to multiple distinct documents or centroids that could in turn be very dissimilar from each other.In STC, as documents may share more than one phrase with other documents, each Document might appear in a number of base clusters. Therefore a document can appear in more than one cluster. Note that the overlap between clusters cannot be too high otherwise they would have been merged into a single cluster.

Features of STC

- Overlapping clusters
- Non-exhaustive
- Linear time
- High precision

2) K - MEANS ALGORITHM

This algorithm is based on partitioning method. Given a database of n objects, and k the Number of clusters to from, a partitioning algorithm organizes the object into k partitions (k <=n), where each partition represents a cluster. The clusters are formed to optimize Objective – partitioning criterion, often called similarity function, such as a distance, so That the objects within a cluster are "similar", whereas the objects of different clusters are "dissimilar" in terms of the database attributes.

The k-means algorithm takes the input parameter k and partitions the set on n objects into K clusters so that resulting intra cluster similarity is high but inter cluster similarity i slow. Cluster similarity is measured in regard to the mean value of the objects in a cluster, which can be viewed as the cluster's center of gravity.

3) AGGLOMERATIVE HIERARCHICAL CLUSTERING ALGORITHM

A hierarchical method creates a hierarchical decomposition of the given set of data Objects. A hierarchical clustering method words by grouping data objects into a tree of Clusters. A hierarchical method can be classified as being either agglomerative or Divisive, based on how the hierarchical decomposition is formed in a bottom-up or top- Down fashion.

Agglomerative Hierarchical Clustering

This bottom-up strategy start by placing each object in its own cluster and then merges These atomic clusters into larger and larger clusters, until all of the objects are in a single Cluster or until certain termination conditions are satisfied.

Divisive Hierarchical Clustering

This top-down strategy does the reverse of agglomerative hierarchical clustering by Starting with all the objects in one cluster. It subdivide the cluster into smaller and smaller Pieces, until each object forms a cluster on its own or until it satisfies certain termination Conditions, such as a desired number of clusters are obtained or the distance between the two closest clusters is a above a certain threshold distance.

4) FRACTIONATION

Fractionation was originally presented by Cutting, Karger, Pedersan and Tukey as a method or extending O (n^2) hierarchical clustering methods to large datasets. In their application the desired number G of clusters was specified a prior; there was no attempt at estimating the number of groups in the data. Let M be the largest number of items to which reasonably apply the base hierarchical clustering procedure. The original Fractionation algorithm proceeds as follows.

1. Split the data into subsets or fractions of size M
2. Cluster each fraction into a fixed number αM of clusters, with α<1. Summarize each cluster by its mean. Refer to these cluster means as meta-observations.
3. If the total number of meat-observations is greater than M, return to step (1). With The meta-observations taking the place of the original data.
4. Cluster the meta-observations into G clusters
5. Assign each individual to the cluster with the closest mean

The number of fractions in the ith iteration is $\alpha n^{1-1} / M$ and the work involved in clustering A fraction is $O(M^2)$ independent of n . This shows that the total run time is linear in n and decreasing in α.

5) SEMANTIC HIERARCHICAL ONLINE CLUSTERING

This algorithm focuses on clustering web search results in order to help users find relevant web information more easily and quickly. The main contributions of this algorithm include the following.

1. The benefits of using key phrases as natural language information features are Discussed. An effect method based on suffix for nay phrase discovery is presented. The efficiency of this method is very high no matter how large the language's alphabet is.
2. The concept of orthogonal clustering is proposed for general clustering problems.

6) COMPARATIVE ANALYSES

The main purpose of this work is to identify the efficient clustering algorithm among the following web document clustering algorithms STC, Fractionation, SHOC, AHC and K-means

The following tables show the comparative results. Here two parameters are used for comparison. The first parameter time is used to mention the time (in seconds) taken for fulfill the query given by the particular clustering algorithm. The second parameter denotes number of document retrieved from each clustering algorithm. There are two types of documents are considered.

(1) Relevant document: the content of the retrieved document is related to the query given by the user.
(2) Irrelevant document: the content of the retrieved document is not related to the query given by the user.

For the above two year types of document the parameters time and number of documents retrieved are noted. The average precision is calculated from the table values using the formula.

$$\text{Average Precision} = \frac{\text{The number of correct answers}}{\text{The total number of answers}}$$

In this experiment, apply various clustering algorithms, to the document collections and compared their effectiveness for information retrieval. The total number of document is 200000000. From this document collection various algorithms are applied.

6.1 single word

First consider the searching keyword as a single word. The searching keyword is "mouse". The following table shows that STC algorithm retrieves more number of related documents with less time compared with other algorithms.

Algorithm	Relevant	Irrelevant	Time	Average precision
STC	1981000	6284500	0.5	0.31
AHC	250000	10900000	0.8	0.02
K-means	1980000	12100000	0.14	0.16
Fractionation	27	122	0.12	0.22
SHOC	1610000	120000000	0.34	0.13

From the above table it is been clearly proved that the STC (Suffix Tree Clustering) algorithm is more efficient and time saving when compared with AHC, K-means, Fractionation and SHOC algorithms.

6.2 Multiple Word

Next consider the searching keyword as a multiple word phrase. The searching keyword is "mouse computer". The documents are retrieved with the words mouse and computer. Comparing the algorithms k-means retrieved more number of relevant documents (1310000) with 0.41 seconds. This shows the average precision for K- means algorithm is high.

The following table shows the results for multiple word phrases

Algorithm	Relevant	Irrelevant	Time	Average precision
STC	1300000	2409480	0.17	0.53
AHC	57900	2440000	0.4	0.02
K-means	1310000	2320000	0.41	0.56
Fractionation	52	112	0.6	0.46
SHOC	1370000	2730000	0.29	0.50

6.3 Overlap

Since documents have multiple topics, it is important to avoid confining each document to only one cluster. That is a good clustering algorithm return a document in more than one cluster. This concept is called **overlap**, which means overlap mechanism allows the documents in more than one cluster.

The table shows that, if the clustering algorithm supports overlapping mechanism then it retrieves more number of documents. The searching time is high compared with no overlap because it search all the clusters where the cluster containing a given keyword or not. The STC shows small difference when overlapping property is applied.

Algorithm	Overlap	No overlap	Time
STC	169	168	0.16
AHC	1420000	1330000	0.33
K-means	1400000	1310000	0.46
Fractionation	55	52	0.9
SHOC	2600000	2730000	0.57

6.4 Performance

The table shows the average time in seconds taken by each service to fulfill a query and the time that the service would time out, or fail to return any hits under one minute. The given query will fail because of many reasons like spelling mistakes and etc.., The keyword given for success is "mouse"-"rat"-"rodent"+"computer".

It means that the document contains only the phrases mouse and computer not rat or rodent. The keyword given for error is "mouse-rat-rodent computer". Here the clustering algorithms consider the keyword as an arithmetic expression and the given query would time out.

Algorithm	success	Time
STC	2134000	0.31
AHC	1100000	0.25
K-means	1070000	0.49
Fractionation	40	0.52
SHOC	1170000	0.44

6.5 Analysis Time

The analysis time the algorithms are given below.

STC - O(n)
K-means - O(nkt)
AHS - $O(n^2)$
Fractionation - $O(n^2)$
SHOC - $O(n^2)$

The above comparisons are found in the following system configuration.

Processor - celeron 1.1 GHz
Hard disk - 20 GB
RAM - 128 MB
Monitor - 15" Samsung
Keyboard - samsung 104
Mouse - mercury
Net connection - DSLleased
Modem - 64 kbps disnet
System - 6 systems are connected at a time

7) CONCLUSION

This work mainly concentrates on identifying the efficient web document clustering algorithms, which are used to retrieve the knowledge from very large databases. The algorithms taken for comparison are STC (Suffix Tree Clustering), K-means, Fractionation, AHC (Agglomerative Hierarchical Clustering) and SHOC (Semantic Hierarchical Online Clustering).The table shows the comparison of different algorithms by using different snippets as the keyword for the document retrieval.The study about the various clustering algorithms in the data mining is applied for comparative analysis. From this work, the STC algorithm shows good results. The STC algorithm implemented in the search engine www.Metacrawler.Com. In future the statistical behavior of various clustering algorithms can be included and the comparative analysis can be extended in practical.

REFERENCES

[1]. Web Document Clustering: Oren Zamir and Oren Etzioni, Dept. of Computer Science and Engineering, University of Washington {Zamir, Etzioni} @cs.washington. edu ACM- 1998

[2]. Projections for Efficient document clustering:H.Schiitze and C.Silverstein. In Proceedings of the 20th International ACM SIGIR Conference on Research and Development in Information Retrieval, 1998

[3]. Multi-service search and comparison using the Metacrawler :E.Selberg and O.Etzioni , proceedings of the 4th World Wide Web Conference, 1995

[4]. Almost-constant time clustering of arbitrary corpus subsets C.Silverstien and J.O.Pedersen. In proceedings of the 20th international ACM SIGIR Conference on Research and Development in Information Retrieval, 1997

[5]. Data mining an overview: Prabhas Chongstitvatana lecturer to master degree students of the computer science department, faculty of science, kasetsartUniversity, 19th July 2002

[6]. Data mining and knowledge discovery: making sense out of data:UsamaM.Fayyad, Microsoft Research – IEEE Expert 1996

[7]. Reality check for data mining:EvangelosSimoudis, IBM Alamaden Research Center, 1996,IEEE Expert

[8]. Datamining: Pieter Adirianns, DolfZantigue – 1999 Addision Wesley

[9]. Datamining: Concepts and Techniques – Jiaweihan, Michelinekamber, Simon Fraser University – 2001, Academic Press

[10]. Web Mining: Clustering Web Documents A Preminary Review:KhaledM.Hammouda, Feb 26,2001

[11]. www.findfast.com

[12]. www.google.com

[13]. www.altavista.com

[14]. www.infoseek.com