EFFICIENT DOCUMENT CLUSTERING USING SUFFIX ARRAY

KOVIT KARAWATREEDECH

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR
THE DEGREE OF MASTER OF SCIENCE
(COMPUTER SCIENCE)
FACULTY OF GRADUATE STUDIES
MAHIDOL UNIVERSITY
2005

ISBN 974-04-5658-8

Thesis
Entitled

EFFICIENT DOCUMENT CLUSTERING USING SUFFIX ARRAY

…….
Mr. Kovit Karawatreedech
Candidate

…….
Assoc.Prof. Damras Wongsawang, Ph.D.
Major-Advisor

…….
Asst.Prof. Chomtip Pornpanomchai, Ph.D.
Co-Advisor

…….
Lect. Sudsanguan Ngamsuriyaroj, Ph.D.
Co-Advisor

…….
Assoc.Prof. Rassmidara Hoonsawat, Ph.D.
Dean
Faculty of Graduate Studies

…….
Assoc.Prof. Supachai Tangwongsan, Ph.D.
Chair
Master of Science Programme
in Computer Science
Faculty of Science

Thesis
Entitled

EFFICIENT DOCUMENT CLUSTERING USING SUFFIX ARRAY

was submitted to the Faculty of Graduate Studies, Mahidol University
for the degree of Master of Science (Computer Science)
on
1 March, 2005

…….
Mr. Kovit Karawatreedech
Candidate

…….
Assoc.Prof. Damras Wongsawang, Ph.D.
Chair

…….
Asst.Prof. Chomtip Pornpanomchai, Ph.D.
Member

…….
Lect. Sudsanguan Ngamsuriyaroj, Ph.D.
Member

…….
Lect. Thepchai Supnithi, Ph.D.
Member

…….
Assoc.Prof. Rassmidara Hoonsawat, Ph.D.
Dean
Faculty of Graduate Studies
Mahidol University

…….
Prof. Amaret Bhumiratana, Ph.D.
Dean
Faculty of Science
Mahidol University

ACKNOWLEDGEMENT

I am enormously grateful to Dr. Damras Wongsawang, my thesis advisor, for his advice, generosity, encouragement and constructive guidance which enable me to fulfill this thesis.

I am also very grateful to Dr. Sudsanguan Ngamsuriyaroj, Dr. Chomtip Pornpanomchai and Dr. Thepchai Supnithi, my thesis supervisory committee members, for their valuable suggestion and their attention.

I wish to thank the Department of Computer Science of Mahidol University for providing the excellent facilities and library, and other graduate friends for their valuable advice and kindness during my study.

I would like to thank my parents for their encouragement and enlightenment which have inspired me to achieve my goal.

Lastly, I also would like to thank all other people who gave me physical and mental support so that I can complete this thesis and my graduate degree.

Kovit Karawatreedech

EFFICIENT DOCUMENT CLUSTERING USING SUFFIX ARRAY

KOVIT KARAWATREEDECH 4237680 SCCS/M

M.Sc. (COMPUTER SCIENCE)

THESIS ADVISORS: DAMRAS WONGSAWANG, Ph.D., SUDSANGUAN NGAMSURIYAROJ, Ph.D., CHOMTIP PORNPANOMCHAI, Ph.D.

ABSTRACT

Nowadays the Search Engine is popular among users in searching the information on the Internet. However, the results are sometimes unmatched with users' need or placed in the latter pages of the document, and the searching wastes their time to arrive at. Such problems have been solved by an idea of clustering that is grouping documents having the same or similar content together. A Suffix Tree Clustering (STC), one of the most popular clustering techniques currently in use, is an efficient technique, which employs the Suffix Tree Algorithm in performing a string matching. The shortfall of the technique is that it requires an extensive memory to execute and due to the complexity of the implementation, some errors may occur. This thesis solves these technical problems by using Suffix Array instead of Suffix Tree.

Similar to the Suffix Tree Clustering technique, the Suffix Array Clustering technique employs a Suffix Array Algorithm in doing a string matching. From our intensive study and program development for the Suffix Array Algorithm and Suffix Tree Algorithm, we found that the major advantages of the Suffix Array Algorithm over the Suffix Tree Algorithm were that the Suffix Array Algorithm is easier to implement and generally requires less memory. In addition, we compared the speed of execution of the two techniques and found that the Suffix Tree Clustering is faster when applied to a small document collection, but slower than, the Suffix Array Clustering when applied to a large document collection. The rationale is that the STC requires more memory space than SAC in maintaining its structure. Thus in the case of collection grows larger, the size of tree grows up faster then that of array. This results in more time taken in maintaining the structure when the tree structure goes beyond the capacity of memory. Hence, SAC is suitable for a large document collection or suitable to run on the computer system with very limited memory resource.

KEY WORDS: DOCUMENT CLUSTERING / SUFFIX TREE / SUFFIX ARRAY

182 P. ISBN 974-04-5658-8

การจัดกลุ่มเอกสารอย่างมีประสิทธิภาพ (EFFICIENT DOCUMENT CLUSTERING USING SUFFIX ARRAY)

โกวิท คารวะตรีเดช 4237680 SCCS/M

วท.ม. (วิทยาการคอมพิวเตอร์)

คณะกรรมการควบคุมวิทยานิพนธ์ : ดำรัส วงศ์สว่าง, Ph.D., สุดสงวน งามสุริยโรจน์, Ph.D., ชมทิพ พรพนมชัย, Ph.D.

บทคัดย่อ

ปัจจุบันการใช้เครื่องมือค้นหาข้อมูลบน Internet โดยใช้ Search Engine ได้รับความนิยมเป็นอย่างมาก แต่ทว่าผลลัพธ์ที่ได้รับจาก Search engine บางครั้งก็ไม่ตรงกับที่ผู้ใช้ต้องการหรือบางครั้งข้อมูลที่ต้องการค้นหาอาจจะอยู่ใน Page หลัง ๆ ไม่ได้อยู่ใน Page แรก ๆ ของเอกสาร ทำให้ผู้ใช้ต้องเสียเวลาในการเข้าถึงข้อมูลที่ต้องการค้นหา จากปัญหาที่เกิดขึ้น โดยการหาวิธีที่จะทำการจัด กลุ่มของเอกสารที่มีเนื้อหาเหมือนกันหรือใกล้เคียงกันจัดให้อยู่ในกลุ่มเดียวกัน Suffix Tree Clustering เป็นเทคนิคที่ดีเป็นที่นิยมใช้ กันในปัจจุบันและมีประสิทธิภาพ โดยวิธีการนี้ใช้ Suffix Tree Algorithm ในการหา String Matching แต่เนื่องจากวิธีนี้ใช้ Memory ค่อนข้างสูงในการทำงานรวมไปถึงเป็นวิธี ที่ค่อนข้างซับซ้อนในการ Implement ซึ่งอาจจะส่งผลให้มีการทำงานผิดพลาดได้ ใน Thesis นี้จึงพยายามที่แก้ปัญหาข้างต้นโดยการใช้เทคนิคของ Suffix Array แทนที่ Suffix Tree

Suffix Tree Algorithm เป็นเทคนิคที่ใช้หา String Matching เช่นเดียวกับเทคนิคของ Suffix Tree clustering จากการศึกษาค้นคว้าของผู้จัดทำพบว่า ข้อดีหลักๆของ Suffix Array Algorithm ก็คือ ง่ายต่อการพัฒนาและยังต้องการขนาดของหน่วยความจำที่น้อยกว่าวิธี Suffix Tree clustering นอกเหนือจากนั้นทางผู้จัดทำยังได้มีการเปรียบเทียบเวลาที่ใช้การประมวลผลของทั้งสองวิธี พบว่า Suffix Tree Clustering นั้นประมวลผลได้เร็วกว่าในกรณีที่ข้อมูลมีขนาดเล็ก แต่จะช้ากว่าในกรณีที่ข้อมูลมีขนาดใหญ่ สาเหตุหลักก็คือ Suffix Tree Clustering ต้องการขนาดของหน่วยความจำที่มากกว่า Suffix Array Algorithm ในการเก็บข้อมูลโครงสร้าง ดังนั้นเมื่อขนาดของข้อมูลใหญ่ขึ้น ขนาดของ Tree ก็จะขยายขึ้นอย่างรวดเร็วกว่า Array ซึ่งก็จะส่งผลให้ใช้เวลามากขึ้นในการรักษาสภาพโครงสร้าง เมื่อโครงสร้างของ Tree มีการใช้เนื้อที่มากกว่าความจุของหน่วยความจำ เนื่องจากเหตุผลดังกล่าว Suffix Array Algorithm จึงเหมาะกับข้อมูลที่มีขนาดใหญ่ หรือ ทำงานบนเครื่องที่มีขนาดของหน่วยความจำที่จำกัด

182 หน้า. ISBN 974-04-5658-8

CONTENTS

CONTENTS (CONT.)

CONTENTS (CONT.)

CONTENTS (CONT.)

LIST OF TABLES

Page

LIST OF TABLES (CONT.)

Page

LIST OF TABLES (CONT.)

Page

LIST OF TABLES (CONT.)

Page

LIST OF FIGURES

Page

LIST OF FIGURES (CONT.)

Page

LIST OF FIGURES (CONT.)

Page

LIST OF FIGURES (CONT.)

Page

CHAPTER I
INTRODUCTION

The rapid growth of the Internet and the World Wide Web (WWW) has changed the way we think and how we seek information. The WWW provides a huge resource in various kinds of topics. There are characteristics of the World Wide Web (WWW) that not only make it extremely powerful, but also present three significant issues to users looking for specific information. Firstly, information sources are distributed all over the world, and are not organized according to any upon indexing scheme. Secondly, the information is heterogeneous and can take many different forms. Thirdly, the WWW is dynamic, with the potential for both content and location to change at any time.

The search tools currently used can be categorized into two groups: Web directories and Search engines [19]. Web directories such as Yahoo give a hierarchical classification of documents; each document in the directory is connected to a node of a tree. Traversing along the tree, a user can access a set of pages that have been manually pre-classified and placed in the tree. Searching in the directory is very convenient and usually leads users to the set of documents they are seeking for. Unfortunately, it leads to only a small fraction of the WWW.

Web searching is based on traditional information retrieval techniques and normally based on Boolean operations of keywords. Major web searching services such as Lycos [15], Alta Vista [16], Yahoo [17] and Google [18] traverse the Web and create an index based on full text of the documents they found. A user enters a keyword query to one of these services, and gets locations of documents found in the index that match the query. Since each service uses different methods in constructing the index, the results of the same queries to different search engines may be different.

Query formats are also different for various services. If a user can skillfully formulate a powerful query, most search engines will retrieve pages with adequate recall, but may be with poor precision [19]. Conventional retrieval methods return the long list of ranked documents that users are forced to scan through to find actual relevant documents. In fact, all current available search tools suffer either from poor precision or poor recall.

The potential and importance of system-aided classification techniques have never been prominent in the increasing presence of digital libraries, online database, and Internet/Intranet applications. The need to have subject-specific structures (taxonomy of categories) for these emerging large-scale information sources, and the difficulty of manually creating and maintaining these structures make classification techniques promising and appealing.

Clustering techniques are unsupervised, and require no external knowledge of categories. Clustering methods try to group similar patterns into the same cluster whose members are more similar to each other (according to some distance measures) than to members of other clusters. There is no priori knowledge of patterns that belong to certain groups, or even how many groups should be formed. Using clustering techniques such as Suffix Tree [6], the results of searching can be organized in various groups of similar subjects, which will greatly improve the information retrieving on the Internet.

1.1 Motivations

Because of very much time consumed in investigating the search results, we have the inspiration to group the similar/related documents into only one group to help users in seeking information. Moreover, the current well-known Document Clustering techniques are quite complex and not efficient in some applications. Therefore, we propose an alternative clustering technique to help improve the performance of document clustering.

1.2 Statement of Problems

Document retrieval systems return a long list of ranked documents that users are forced to scan through to find documents relevant to their queries. On the Web, this problem is escalated by high recall and low precision of search engines such as Alta Vista, Yahoo and Google [20]. Moreover, a typical user has troubles in identifying highly specific queries and does not take advantage of advanced search options.

Document clustering is a method that permits users to efficiently navigate through a large collection of searching results. One of the motivations for using clustering algorithms is to provide automated tools to help in construction categories or taxonomies for user and in searching a large collection of database quickly.

One common problem of clustering methods is the difficulty in interpretation of the clusters. Most clustering algorithms prefer certain cluster shapes, and the algorithms will always assign data to clusters of such shapes even if there is no cluster in the data. Another problem is that the choice of the number of clusters may be critical. Moreover, the appropriate number needed to be assigned for threshold is also a major concern.

This thesis studies and explores various clustering methods a currently in use, and proposes an alternative solution to clustering technique using Suffix Array [21]. The document clustering technique using Suffix Array Clustering, or SAC, is expected to be analyzed both theoretically and experimentally. Its prototype will be developed and tested for performance evaluation in the actual situation of on-line Internet search. Furthermore, a comparison to other clustering techniques will be also presented.

1.3 Objectives of Study

Based on the afore-mentioned problems, the objectives of this research are listed as follows:

1. To study current techniques in Web document clustering.

2. To propose an alternative Web document clustering technique that employs Suffix Array, called SAC (Suffix Array Clustering method).

3. To develop a prototype of SAC for performance evaluation in terms of processing time, memory usage and precision of the clustering.

1.4 Problem Scope

This research has drawn the scope of study and experiment for the sake of feasibility as follows:

1. We will survey and study clustering techniques currently in use such as K-Means, Single-Pass and Suffix Tree Clustering (STC) [6] and bring them to our consideration for comparison.

2. The design and implementation of a SAC prototype will be developed by using a general purpose programming language without any extra development tools.

3. The SAC prototype will be tested on a general PC.

4. A simplified version of Porter's stemming [14] is used.

5. The experiments will be conducted on a test data set gathering from many search engines such as Google, Yahoo, etc. The documents in the data set contain various types of contents.

6. Three evaluation areas will be taken into our consideration for performance measurement, processing time, memory usage and clustering precision.

1.5 Thesis Organization

The thesis is outlined as follows:

Chapter I: Introduction, this chapter explains about the background, the statement of problems, the motivations, the objectives of study, the problem scope and the outline of this thesis.

Chapter II: Literature Review, this chapter reviews and explores the Document Clustering techniques using in the present day and their related works such as Stemming [14].

Chapter III: The approach, this chapter proposes a clustering technique using Suffix Array for effective Document Clustering.

Chapter IV: System Design and Implementation, this chapter presents the detailed design and the model development methods of SAC, which include system overview, system design and implementation.

Chapter V: Experimental Results, this chapter shows the setup of experiments and results of the prototype developed.

Chapter VI: Discussion and Conclusion, this chapter discusses the performance of the proposed clustering technique. Some conclusions are made as well as some further developments are suggested.

CHAPTER II
LITERATURE REVIEW

This chapter reviews some background knowledge on document clustering and also surveys some research works on clustering techniques. The contents include some clustering techniques previously proposed and some techniques currently in use such as Web document clustering, Pairwise Agglomerative Clustering [3], K-Means [8] and Suffix Tree Clustering [6]. Furthermore, some related topics such as information filtering, stopword removal, word stemming are reviewed.

2.1 Document Clustering

Document clustering is the method to group similar documents together. Previously, grouping of the document requires man-power to execute. However, the number of documents on the Internet increases tremendously and man-power resource could not be able to manage it. Then, technology is brought to manage grouping automatically. Document clustering method is different from other clusterings because a document consists of various types of words combining together. Some words have no meaning; some have one meaning, and some have many meanings. The widely used document clustering techniques include K-Means [8], Pair-Wise [3], Single Pass [23], Suffix Tree Clustering [6], which will be later reviewed.

2.2 Clustering Methods

General clustering methods are categorized by using the type of cluster structures they produce [1]. The nonhierarchical method, known as partitioning method, divides the data set of N objects into M clusters but no overlapping is allowed.

Items, which are most similar, are put to the same cluster as a membership of cluster, and the cluster may have a centroid or cluster representative in order to indicate the common characteristics of the items it contains. Another clustering method is more complex, the hierarchical method. It produces a nested data set consisting of pairs of items or clusters successively linked until every item in the data set is connected. The hierarchical method can be either agglomerative, with *N-1* pair wises join starting from an un-clustered data set, or divisive, starting with all objects are put in a single cluster and then progress through *N-1* divisions of some large clusters into smaller clusters. The divisive methods are less commonly used and few algorithms are available, hence only a class of agglomerative methods will be described in this chapter.

2.2.1 Nonhierarchical Methods

The nonhierarchical methods are heuristic in nature [2] because priori decisions about the number of clusters, cluster size, criteria for cluster membership and form of cluster representation are required. As the large amount of possible divisions of N items into M clusters to make the best solution is unlikely, the non-hierarchical methods try to find an approximation, usually by partitioning the data set in some ways and relocating items until criteria are optimized. The computational requirement of $O(NM)$ is much lower than the hierarchical methods if $M << N$. As a result, the large amount of data sets can be partitioned. This type of methods is used for most of previous works in document clustering when computational resources were limited.

2.2.2 Hierarchical Methods

The majority of the early published works on cluster analysis engaged hierarchical methods whereas general methods in information retrieval areas are not. There are many areas having benefits from implementing hierarchical methods including computer resources management and, the availability of software packages for cluster analysis and algorithms [2].

The results from the hierarchical agglomerative clustering methodology are displayed as a dendrogram as shown in Figure 2.7. The dendrogram is a useful representation when considering retrieval from a clustered set of documents, since it shows document clusters and also indicates the paths of retrieval process.

The commonly used hierarchical agglomerative clustering methods and their characteristics are briefly described as follows [2]:

1. **Single Link:** At each step, the Single Link method joins the most similar pair of objects that are not yet put in the same cluster. Theoretically, the Single Link method has some attractive properties and it can be implemented efficiently. However, the Single Link has a structure of long messy clusters, which makes it suitable only for defined relevant clusters but unsuitable for poorly separated clusters.

2. **Complete Link:** Basic idea of this method refers to the linking of all entities in a cluster to one another within some minimum similarities. The Complete Link method also uses the least similar pair between each of two clusters to determine the inter-cluster similarity.

3. **Group Average Link:** The Group Average Link method uses an average value of the pair-wise links within a cluster to determine similarity. All objects also contribute to inter-cluster similarity, resulting in a structure compromising between the loosely bound Single Link cluster and tightly bound Complete Link cluster.

4. **Ward's Method:** The Ward's method joins at each stage the cluster pair whose merger minimizes the increase in the total within-group error sum of squares based on the distance between centroids. By this way, it is also known as the minimum variance method. It also tends to produce homogeneous clusters and a symmetric hierarchy.

Its definition of a cluster center of gravity provides a useful way of representing a cluster. Some tests have shown that the Ward's method is good at recovering cluster structure, though it is sensitive to outliers and poor at recovering elongated clusters.

2.3 Pairwise Agglomerative Clustering

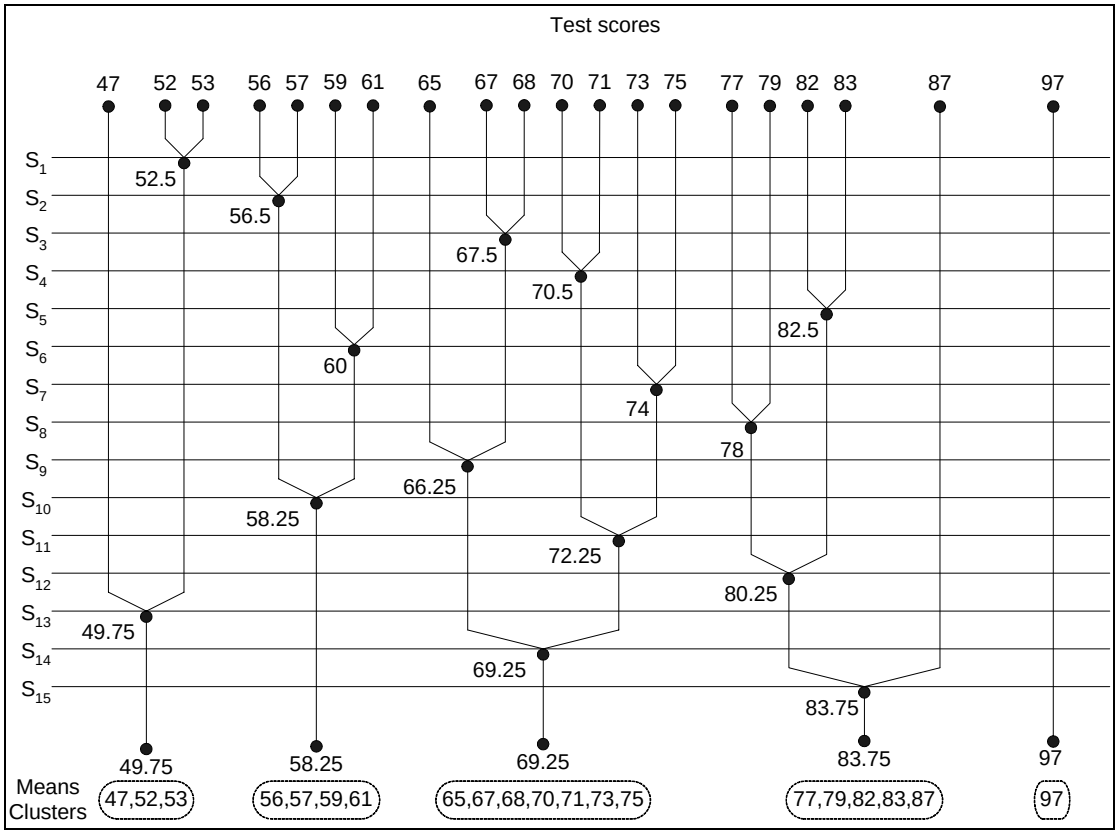


Figure 2.1: The operation of an agglomerative clustering method.

The algorithm steps can be shown in a tree structure. At internal nodes, two branches are merged into one. For example, two data points are merged into one clusters, or two clusters are merged into one larger cluster. In Figure 2.1, there are 20 initial separate clusters which each cluster contains only one data point. Then, they are merged into larger clusters but produce fewer numbers of clusters. Firstly, the two closest points (scores of 52 and 53) are merged into one cluster {52, 53} and is replaced by an average value of the two points (52.5). Next, we repeat the same

process by searching the closest two points, calculating the average value and then merging the points. Therefore, there will be only one new branch at each step.

2.4 K-Means Clustering

The iterative process of K-Means clustering [8] starts from a set of *k* reference points with the initial values chosen by a user. The process begins with a set of data points partitioned into *k* clusters. A data point *x* becomes a member of cluster *i* if z_i, the reference point of cluster *i*, is closest to *x*. The position of the reference points is adjusted during the iterative process. The iterative algorithm uses "guess" method for each fitted parameter and then optimizes their values. The series of this algorithm are different on how to generate and how to adjust the partitions. There are three members of this type of clustering; Lloyd's algorithm, the standard K-Means algorithm, and the continuous K-Means algorithm [8].

2.4.1 Lloyd's Algorithm

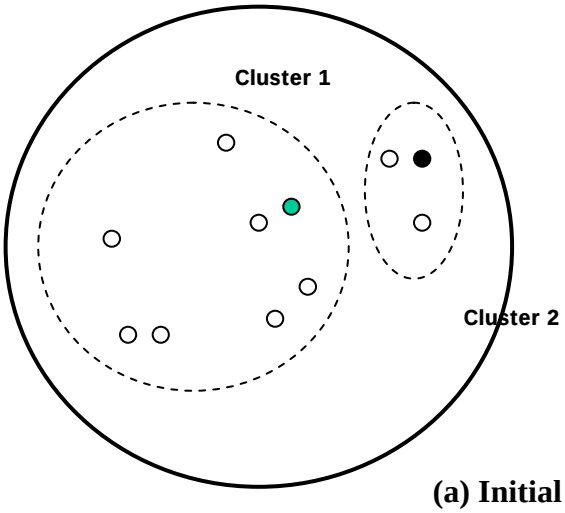
The Lloyd's algorithm [8] is the simplest one. The initial partitioning is set up as all data points are partitioned into *k* clusters by assigning each point to the closest cluster. Then, we re-calculate the centroid values of those clusters and use those centroids as the reference points for the next partitioning of all the data points. It means that each data point in a cluster is closer to the reference point of its cluster than any other point, and this reference point is referred to as the centroid of its cluster.

For Lloyd's and other iterative algorithms, it is quite fast to improve the partitioning although the initial reference points are badly selected. The algorithm can guarantee only the final partitioning will be better than that of the initial selected. However, it will not be necessary the best partitioning.

2.4.2 Standard K-Means Algorithm

Although the steps of the standard K-Means and Lloyd algorithms are almost the same, the standard K-Means has more efficiently use of information at every step. The standard K-Means algorithm begins with choosing a reference point and assigning all data points to clusters. It also uses the cluster centroids as the reference points, but the centroids are adjusted both during and after each partitioning. For data point x in cluster i, if the centroid z_i is the nearest reference point, no adjustment is made and the algorithm proceeds to the next data point. However, if the centroid z_j of the cluster j is the nearest reference point to x, data point x will be moved into cluster j. Then the centroid of the "losing" cluster i (minus point x) and the "gaining" cluster j (plus point x) are recomputed, and the reference points z_i and z_j are moved to their new centroids. After each step, every one of the k reference point is a centroid.

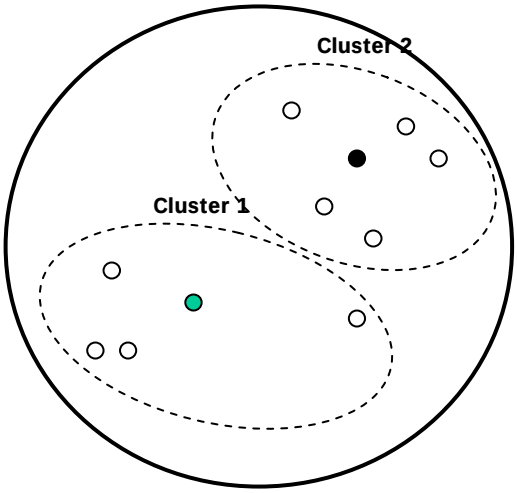
The standard K-Means algorithm [8] is depicted in Figure 2.2. The process in details is described as follows.



(a) Initial clusters.

Figure 2.2 (a) : Clustering by the Standard K-Means algorithm.

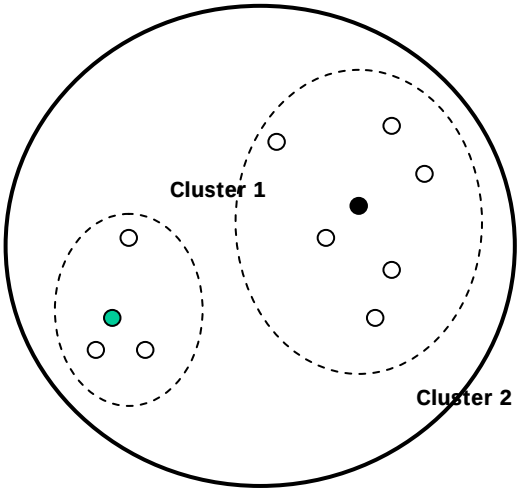
(a) Setup: Reference point 1 (filled SHADED circle) and reference point 2 (filled DARK circle) are chosen arbitrarily. All data points (open circles) are then partitioned into two clusters. Each data point is assigned to cluster 1 or cluster 2 depending on whether the data point is closer to reference point 1 or 2 respectively (Figure 2.2 (a)).



(b) Intermediate clusters.

Figure 2.2 (b) : Clustering by the Standard K-Means algorithm.

(b) Iteration: Next each reference point is moved to the centroid of its cluster. Then each data point is considered in the sequence shown. If the reference point is closest to the data point belonging to the other cluster, that data point is reassigned to that other cluster, and both centroids of clusters having members transferred are recomputed (Figure 2.2 (b)).



(c) Final clusters.

Figure 2.2 (c) : Clustering by the Standard K-Means algorithm.

(c) Finalization: The process in Figure 2.2 (b) is repeated until we reach a stable state; i.e., there is no change in clustering, then, we stop the iteration and get the final clusters (Figure 2.2 (c)).

For the standard K-Means algorithm, reference points are chosen and all the data points are assigned to clusters. As with Lloyd's, the K-Means algorithm uses the cluster centroids as reference points in subsequent partitioning—but the centroids are adjusted both during and after each partitioning. For data point x in cluster i, if the centroid z_i is the nearest reference point, no adjustments are made and the algorithm proceeds to the next data point. However, if it is not, the centroid of the "losing" cluster i (minus point x) and the "gaining" cluster j (plus point x) are recomputed, and the reference points z_i and z_j are moved to their new centroids. After each step, every one of the k reference point is a centroid, or mean, hence this method was named "K-Means".

2.4.3 Continuous K-Means Algorithm

The continuous K-Means algorithm [8] is faster than the standard version. It differs from the standard one on how to select the initial reference points and how to select the data points for the updating process.

In the continuous algorithm, the reference points are chosen randomly from the whole data points. If the sample is large enough, the distribution of the reference points can be the representative of the entire data points. If the densest is in the region *i*, the sample should be densest in the region *i* as well.

Another difference between the standard and the continuous K-Means algorithm is the way they treat their data points. During each iteration, the standard version examines all data points orderly while the continuous algorithm examines only the random sample of data points. If the data set is very large and the sample is the good representative of the data set, the continuous algorithm will be quicker than the standard version.

2.5 Suffix Tree Clustering (STC)

Suffix Tree Clustering (STC) [6] is a linear time clustering algorithm based on identifying the groups of documents. It is designed to meet the requirements of post-retrieval clustering of web search results. A unique phase in context is an order sequence of a string of words. A base cluster or a suffix tree is identified to be a set of documents that share common phases. There are 3 main steps in performing the STC as follows:

1.) **Document Cleaning**

In this step, all stopwords are removed and then the string of each document is transformed using the stemming algorithm. The stemming algorithm is to delete prefix/suffix or, change plural to singular. Moreover, any sentence boundaries (punctuation marks, HTML tags) and any non-words

(numbering, preposition, article) are removed. Only the stemmed strings are kept.

2.) **Identifying Base Clusters**

The identification of base clusters can be viewed as the creation of an inverted of phrases for our document collection. The structure can be constructed in linear time with the size of the collection.

We treat documents as strings of words, not characters, then suffixes contain one or more whole words.

- A suffix tree is a rooted, directed tree.
- Each internal node has at least 2 children.
- Each edge is labeled with non-empty sub-strings of S (S is suffix tree of a string). The label of a node is defined to be the concatenation of the edge-labels on the path from the root to that node.
- No two edges out of the same node can have edge-labels that begin with the same word.
- For each suffix *s* of S, there exists a suffix-node whose label equals *s*.

The tree containing a set of strings is a compact Tries [6] consisting of all the suffixes of all the strings in the collection. Each suffix-node is marked to designate from which string it originated from. An example is as follows:

Suppose that document1, document2 and document3 contain strings "*cat ate cheese*", "*mouse ate cheese too*" and "*cat ate mouse too*" respectively, a step by step creating of suffix tree containing a set of strings "*cat ate cheese*", "*mouse ate cheese too*" and "*cat ate mouse too*" are described as follows.

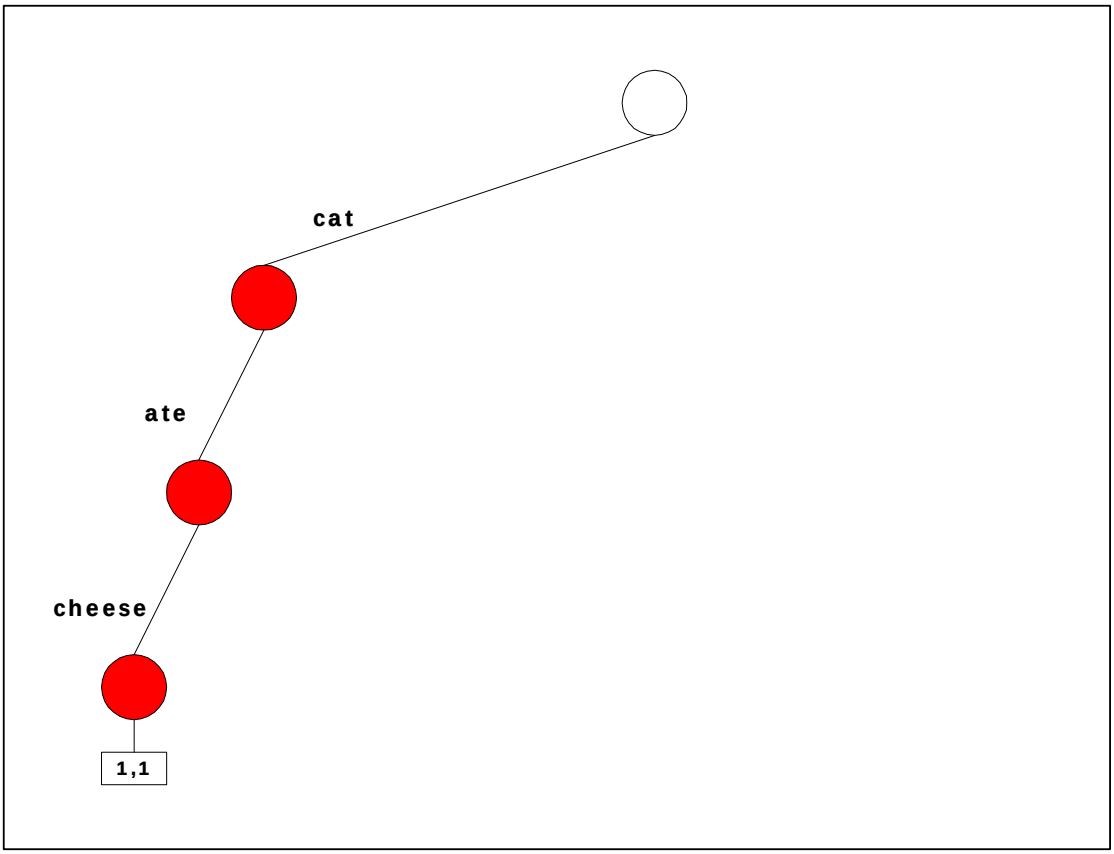
First we parse all three strings into suffixes. A set of suffixes we obtained consists of

Document1	**Document2**	**Document3**
1. "cat ate cheese"	*1. "mouse ate cheese too"*	*1. "cat ate mouse too"*
2. "ate cheese"	*2. "ate cheese too"*	*2. "ate mouse too"*
3. "cheese"	*3. "cheese too"*	*3. "mouse too"*
	4. "too"	*4. "too"*

Next, we sequentially add each suffix to the suffix tree as described in the following steps.

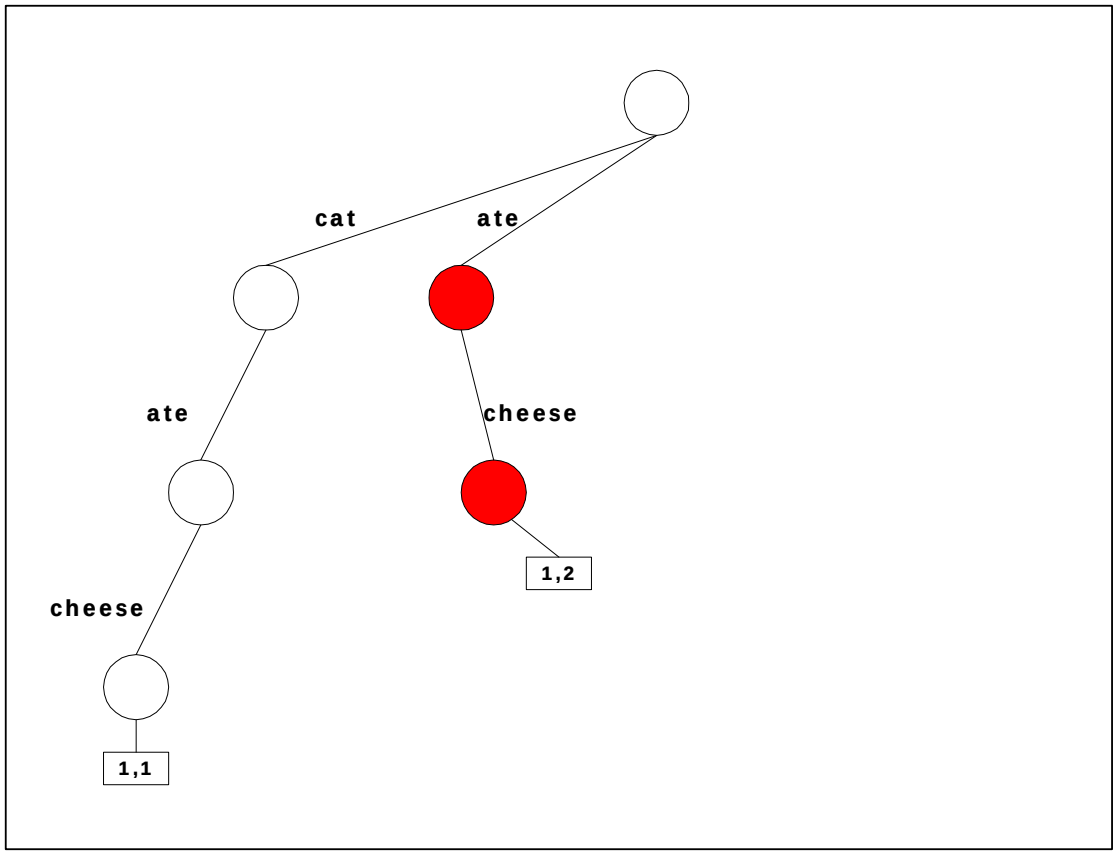
Step0: Starting with a null tree.
Step1: Adding "cat ate cheese" to null tree
Step2: Adding "ate cheese"
Step3: Adding "cheese"
….
:
:
Step11: Adding "too"
Step12: Reduction of sub-tree containing nodes having single child.

A pictorial incremental construction of this suffix tree is shown in Figure 2.3 (a) – Figure 2.3 (k). The pair of numbers shown in a rectangle, (x,y) at each leaf node indicates the document number (x) and the word or phrase number (y) of that document respectively.



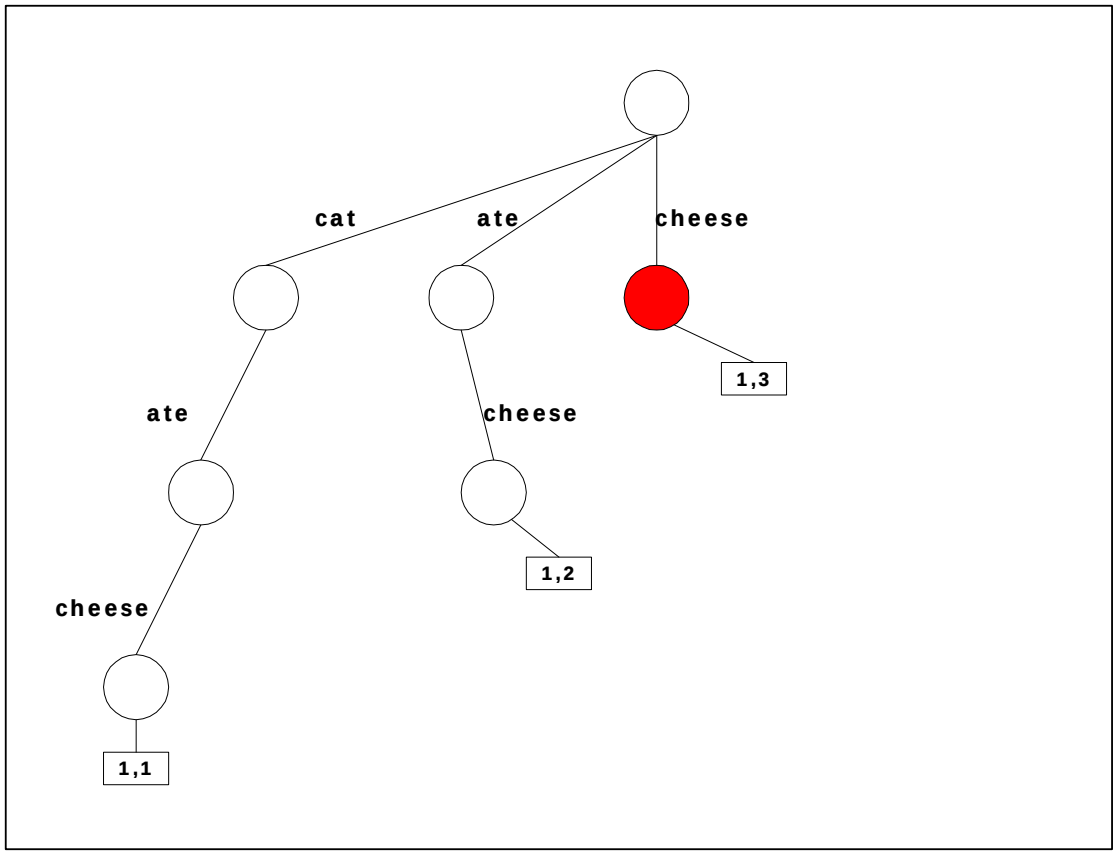
(a) Adding “cat ate cheese”.

Figure 2.3: A step by step suffix tree construction.



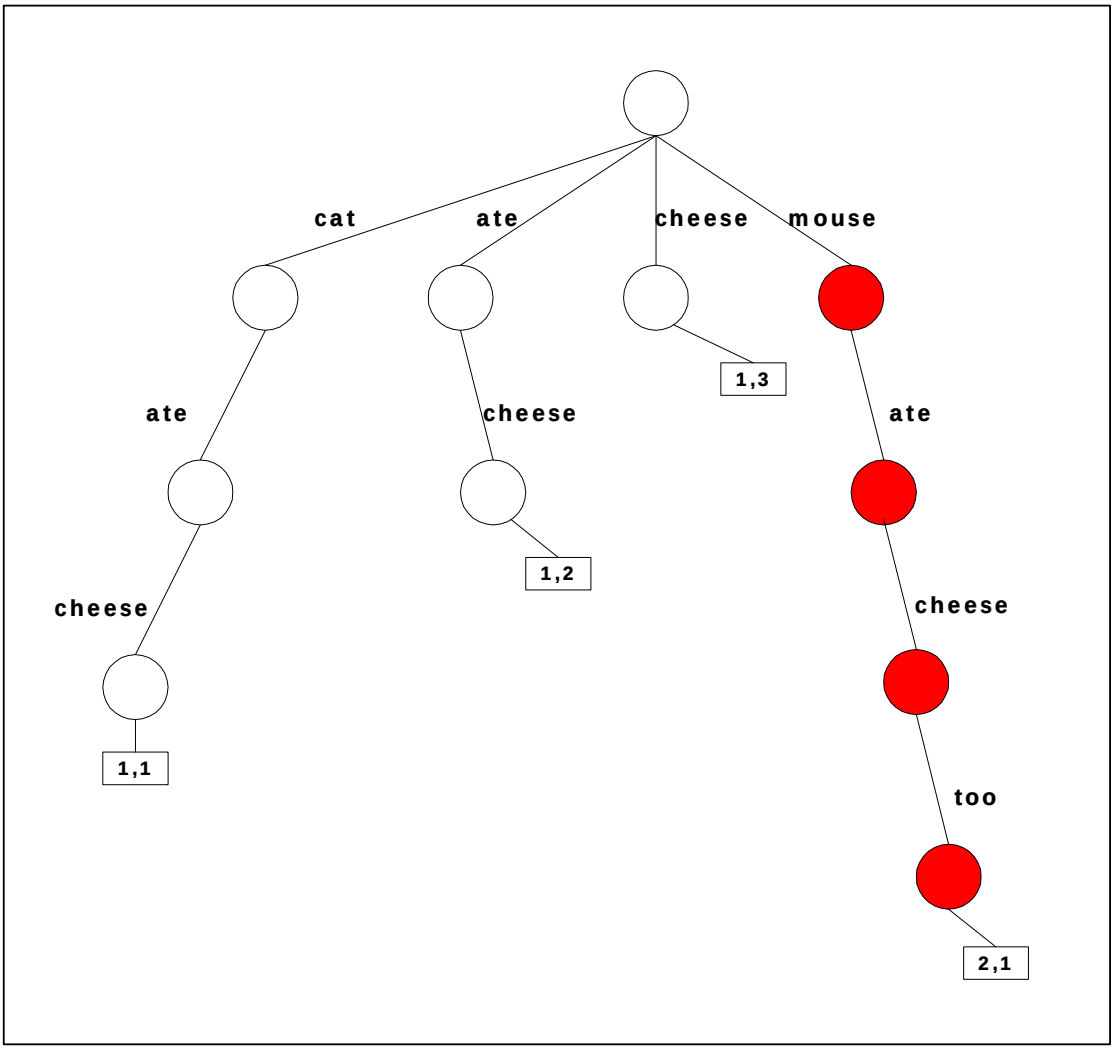
(b) Adding “ate cheese”.

Figure 2.3: A step by step suffix tree construction. (cont.)



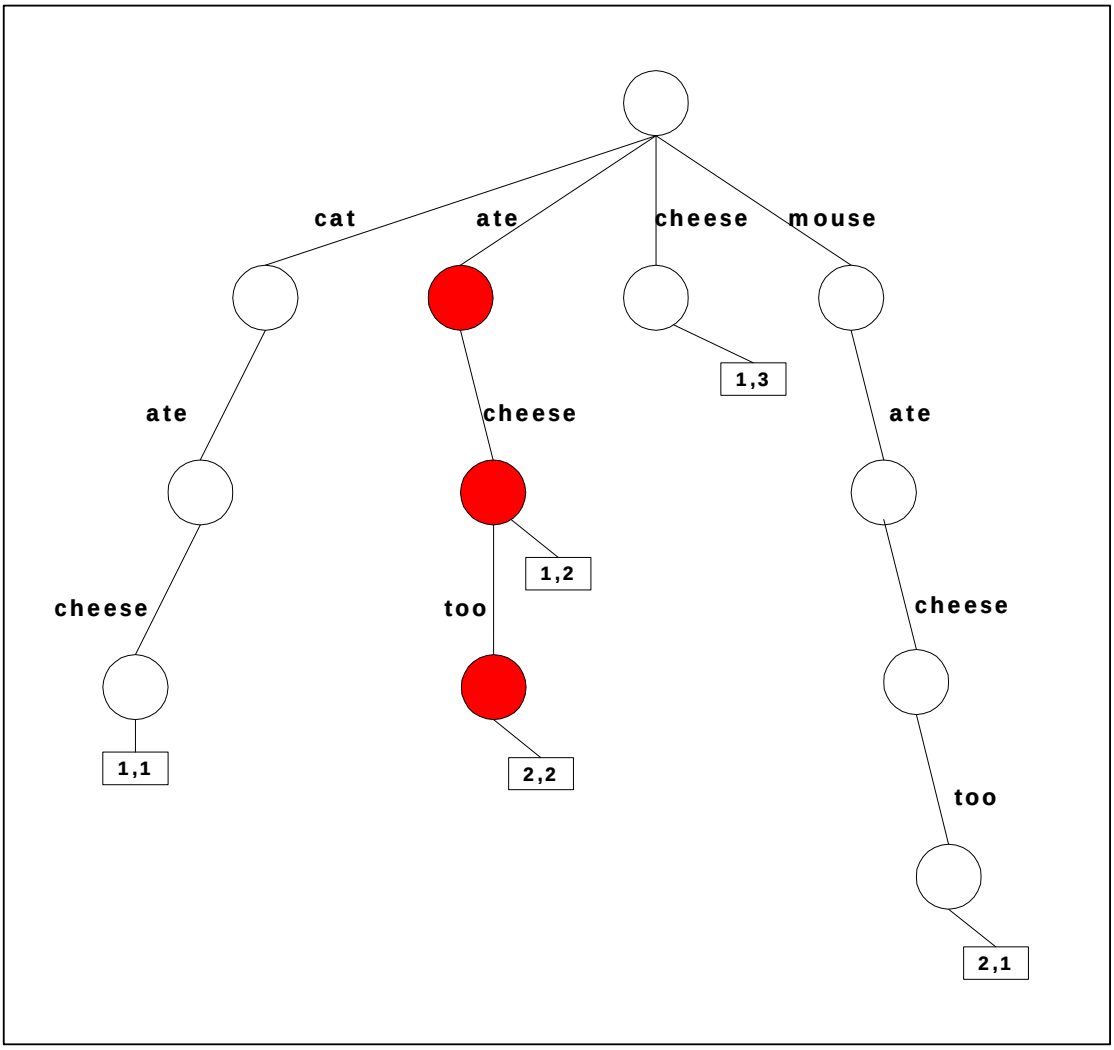
(c) Adding “cheese”.

Figure 2.3: A step by step suffix tree construction. (cont.)



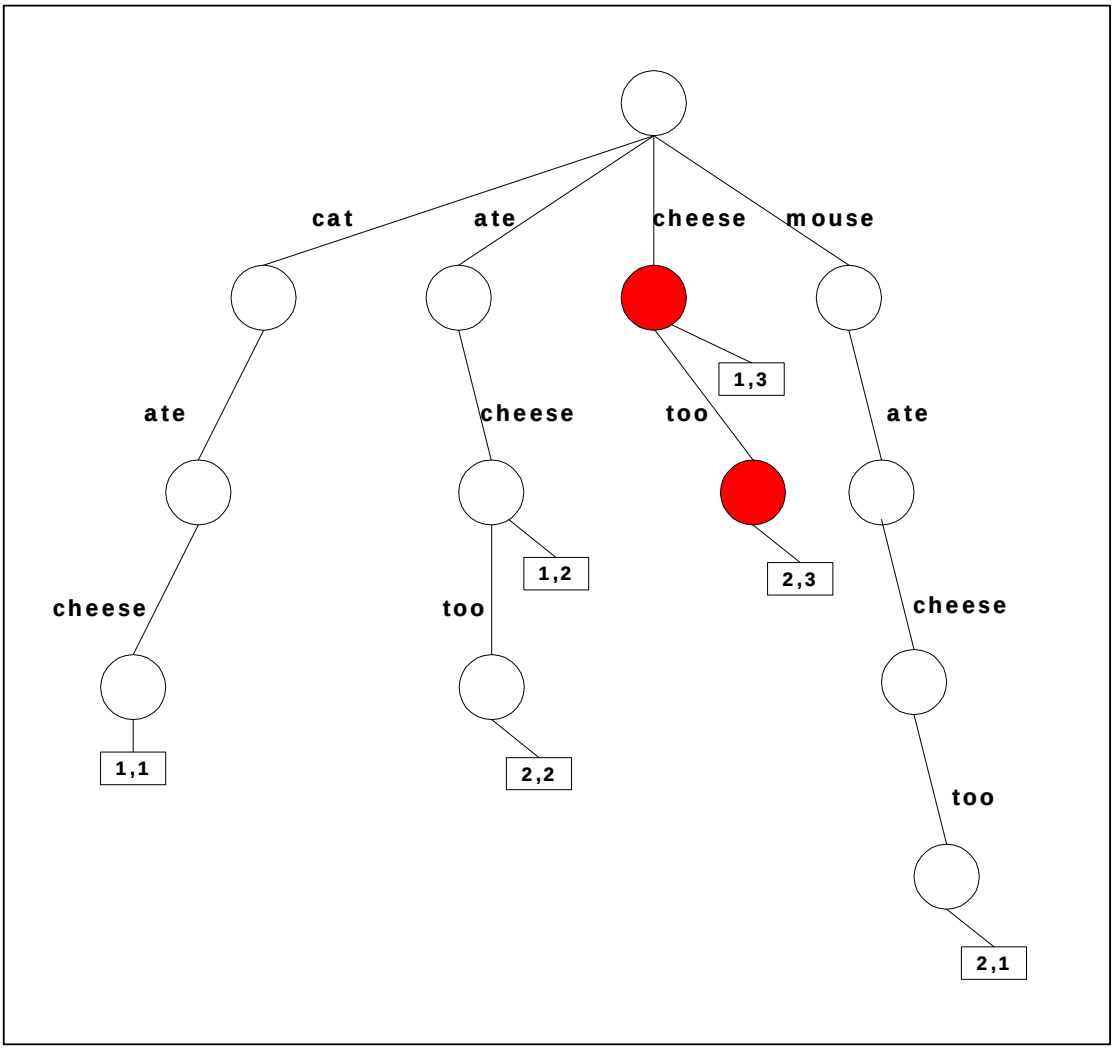
(d) Adding “mouse ate cheese too”.

Figure 2.3: A step by step suffix tree construction. (cont.)



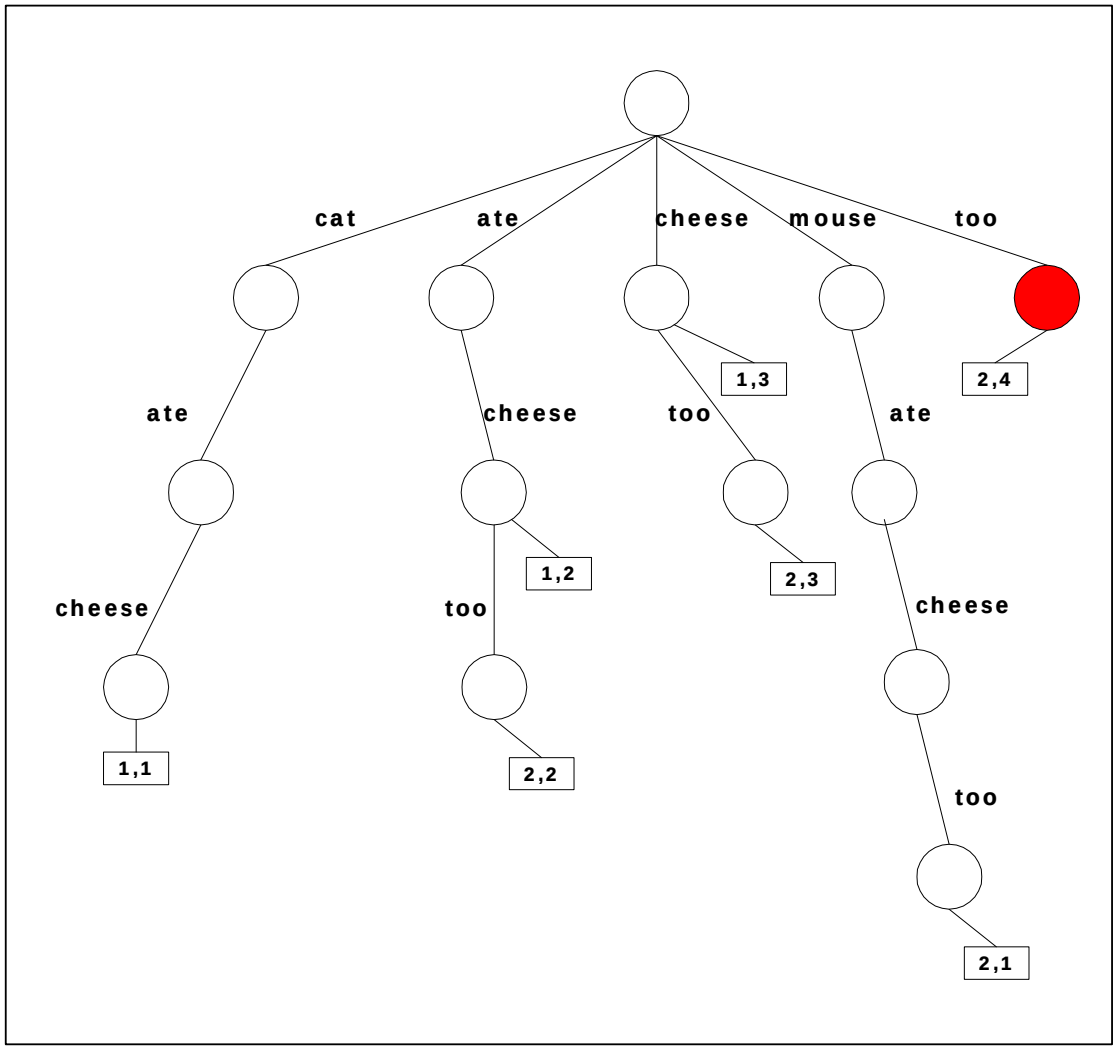
(e) Adding “ate cheese too”.

Figure 2.3: A step by step suffix tree construction. (cont.)



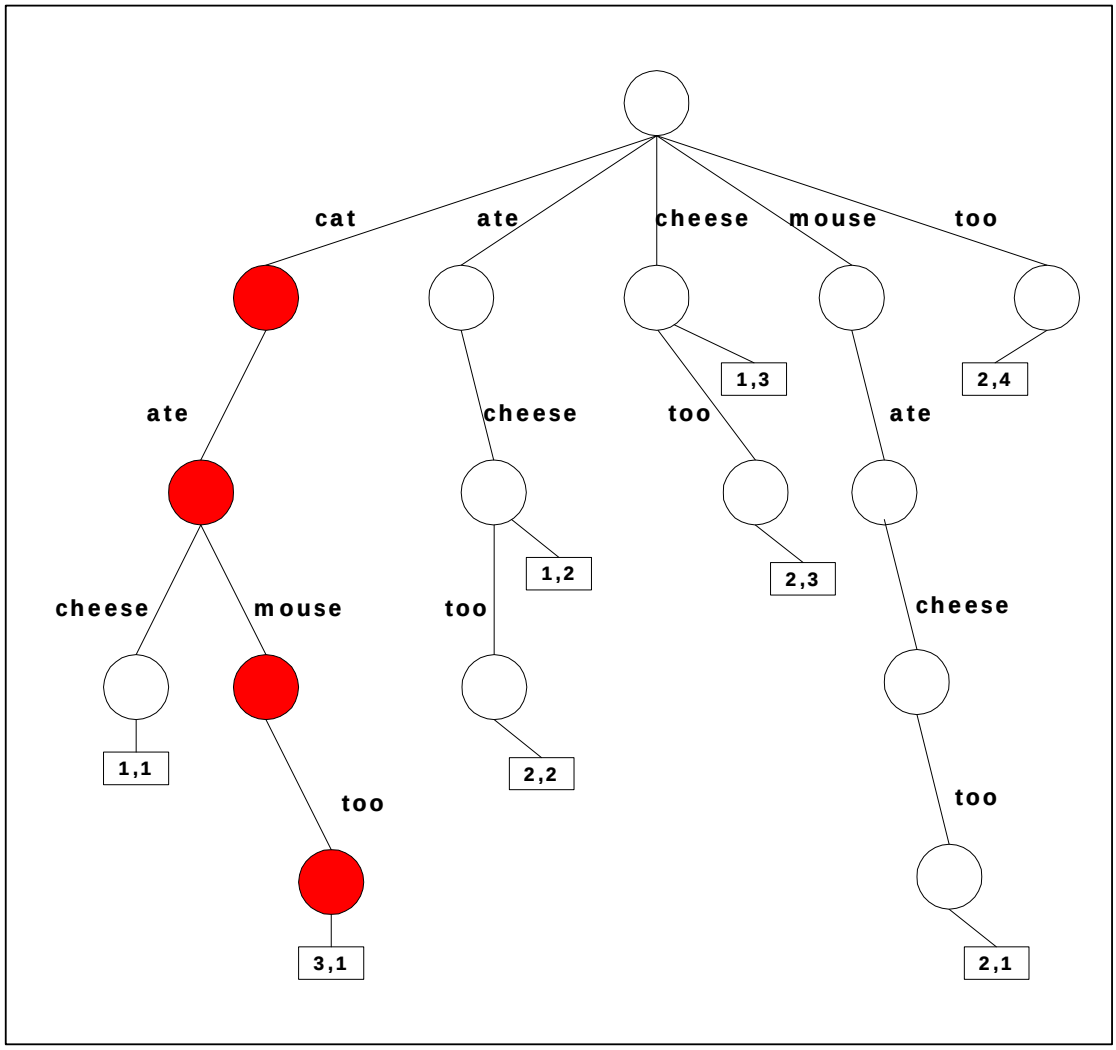
(f) Adding “cheese too”.

Figure 2.3: A step by step suffix tree construction. (cont.)



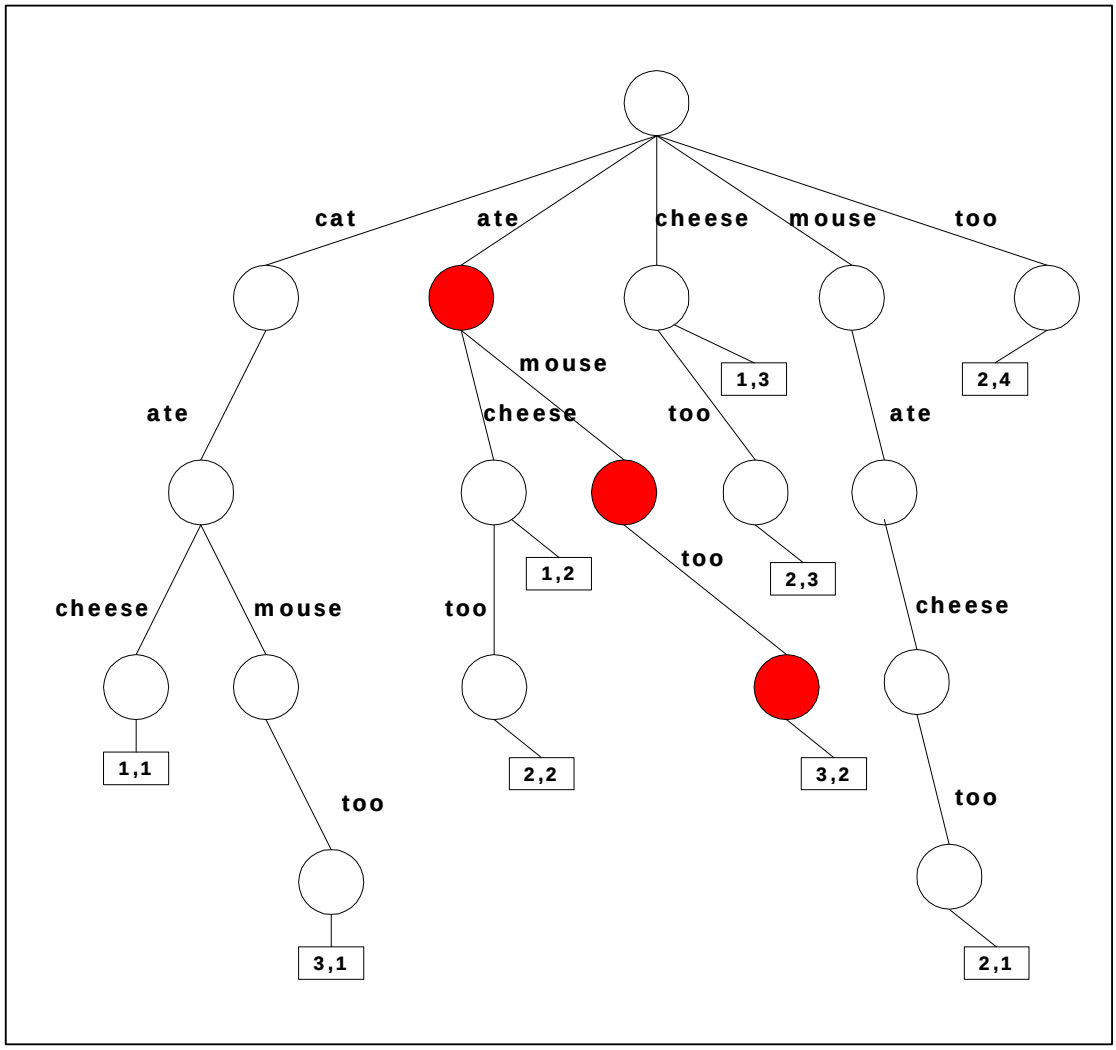
(g) Adding “too”.

Figure 2.3: A step by step suffix tree construction. (cont.)



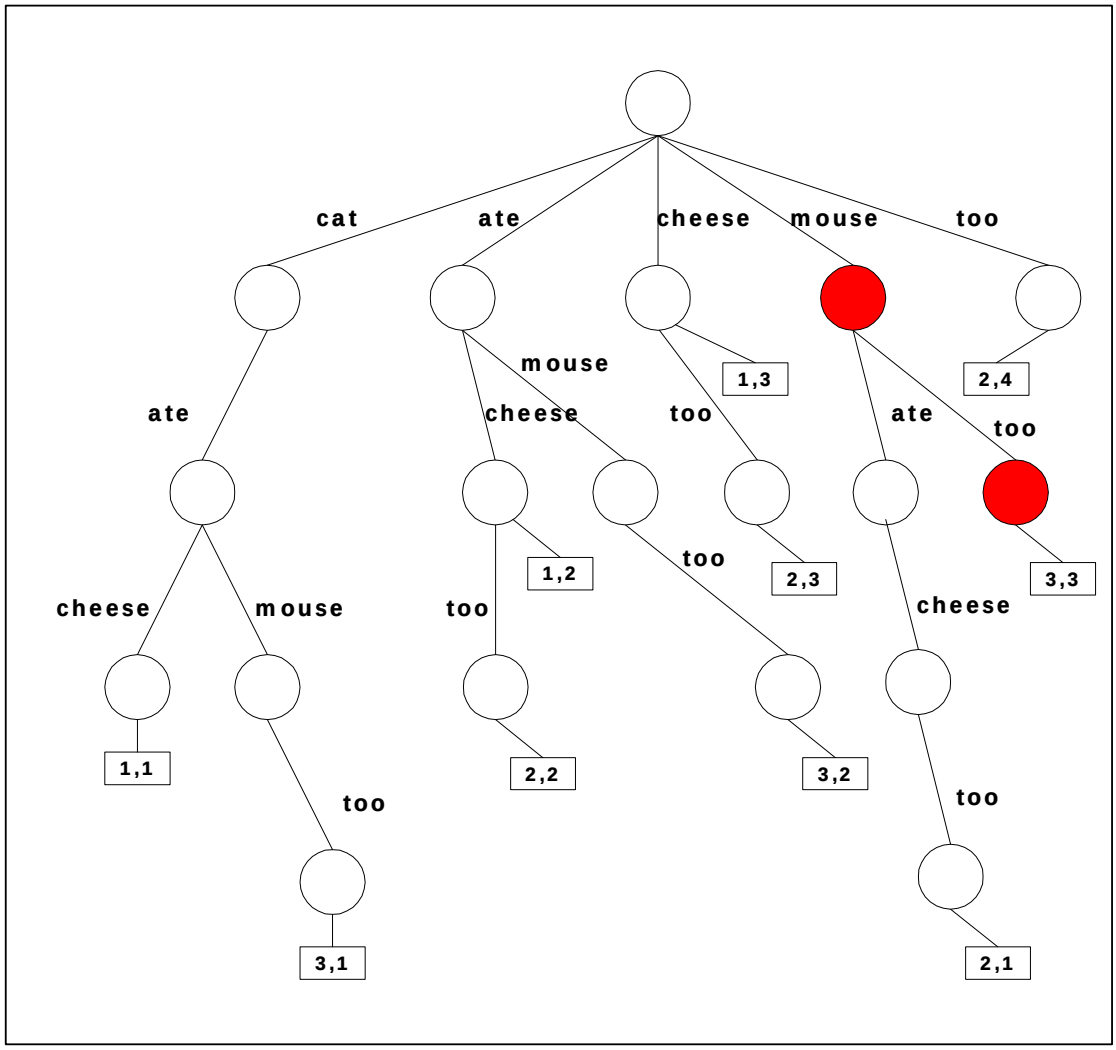
(h) Adding “cat ate mouse too”.

Figure 2.3: A step by step suffix tree construction. (cont.)



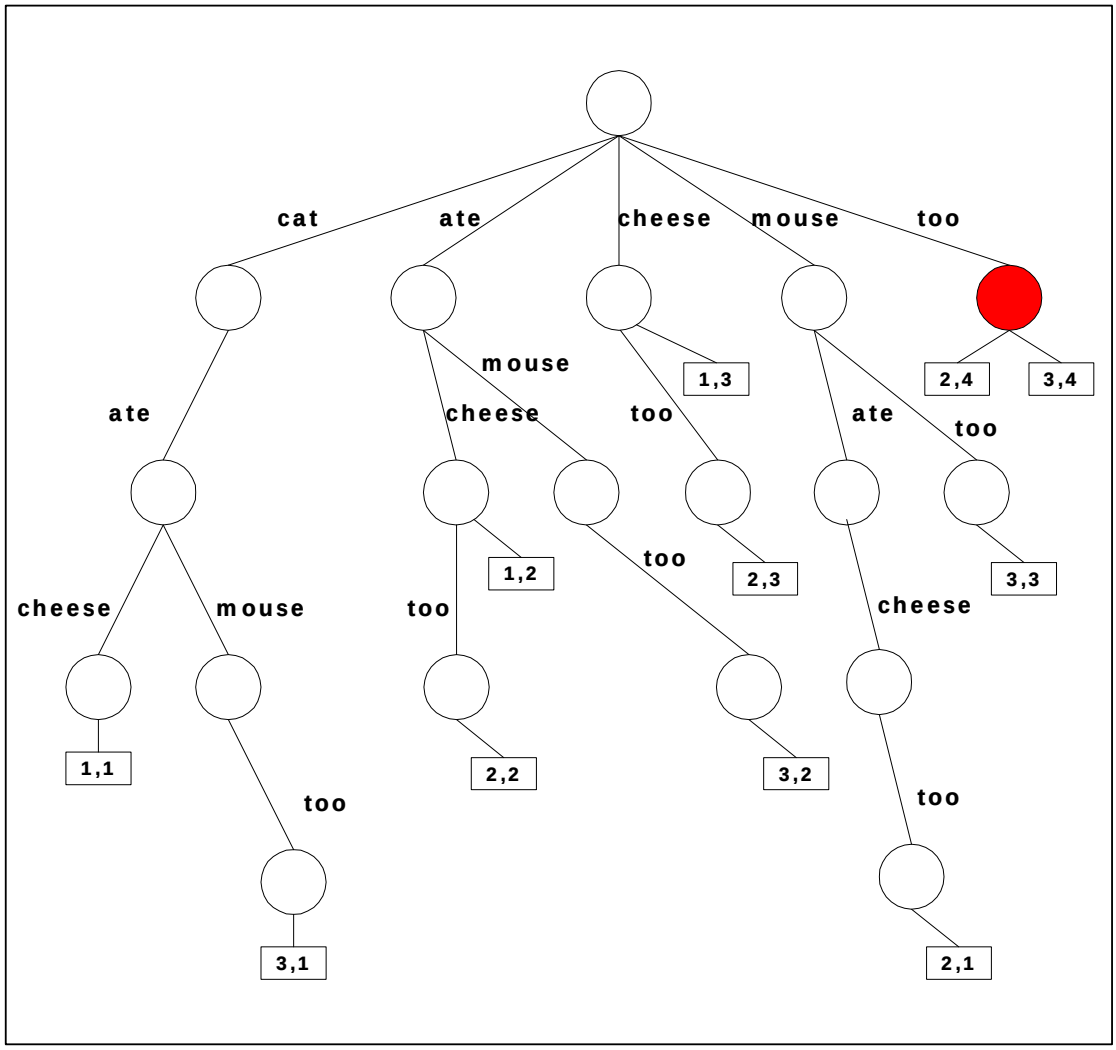
(i) Adding “ate mouse too”.

Figure 2.3: A step by step suffix tree construction. (cont.)



(j) Adding “mouse too”.

Figure 2.3: A step by step suffix tree construction. (cont.)



(k) Adding “too”.

Figure 2.3: A step by step suffix tree construction. (cont.)

After adding all suffixes to the tree, we perform a tree compaction. The tree compaction is done by reduction of sub-tree containing nodes having single child. Figure 2.4 shows nodes of the suffix tree, in shade, having a single child where reduction can be performed.

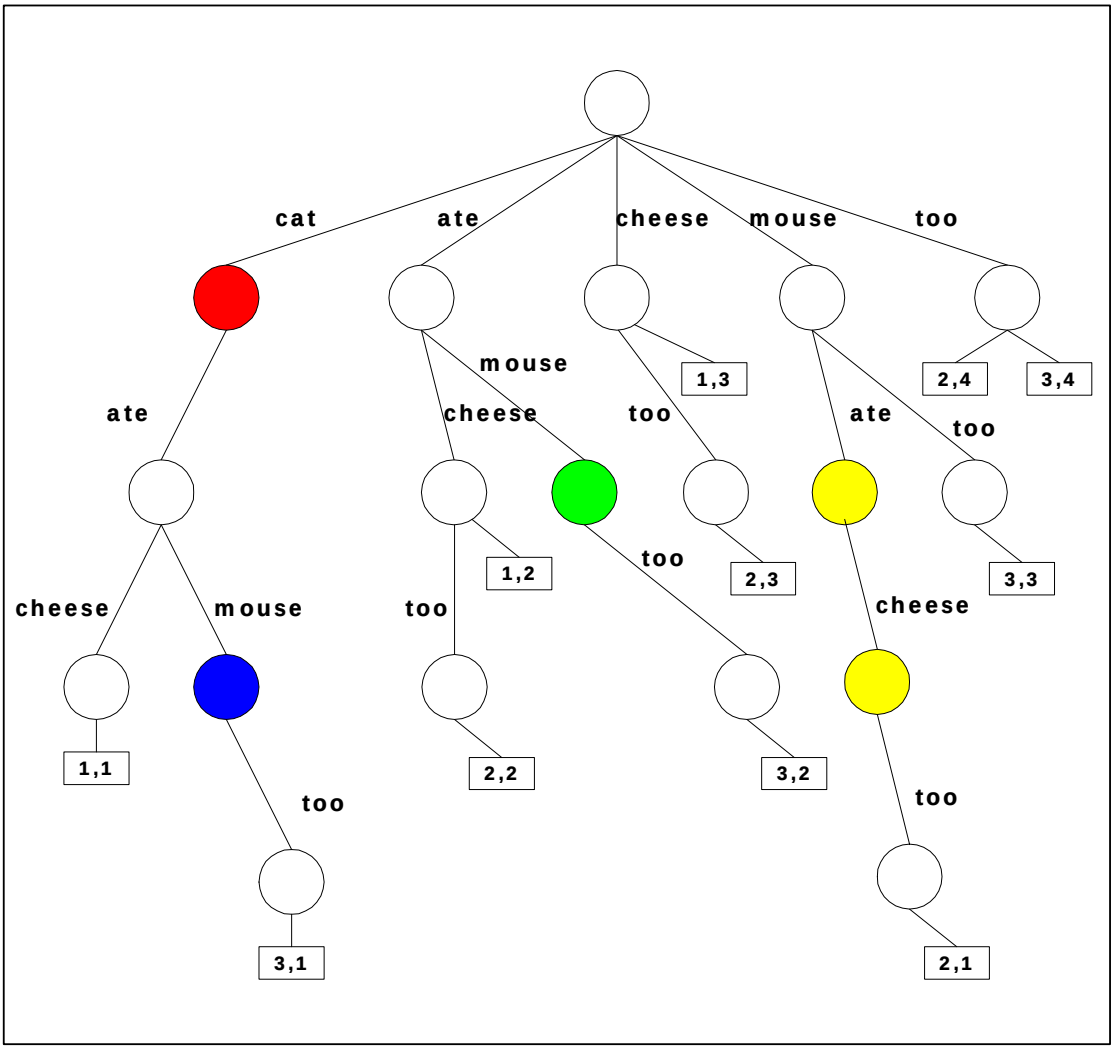


Figure 2.4: Suffix tree showing nodes having single child.

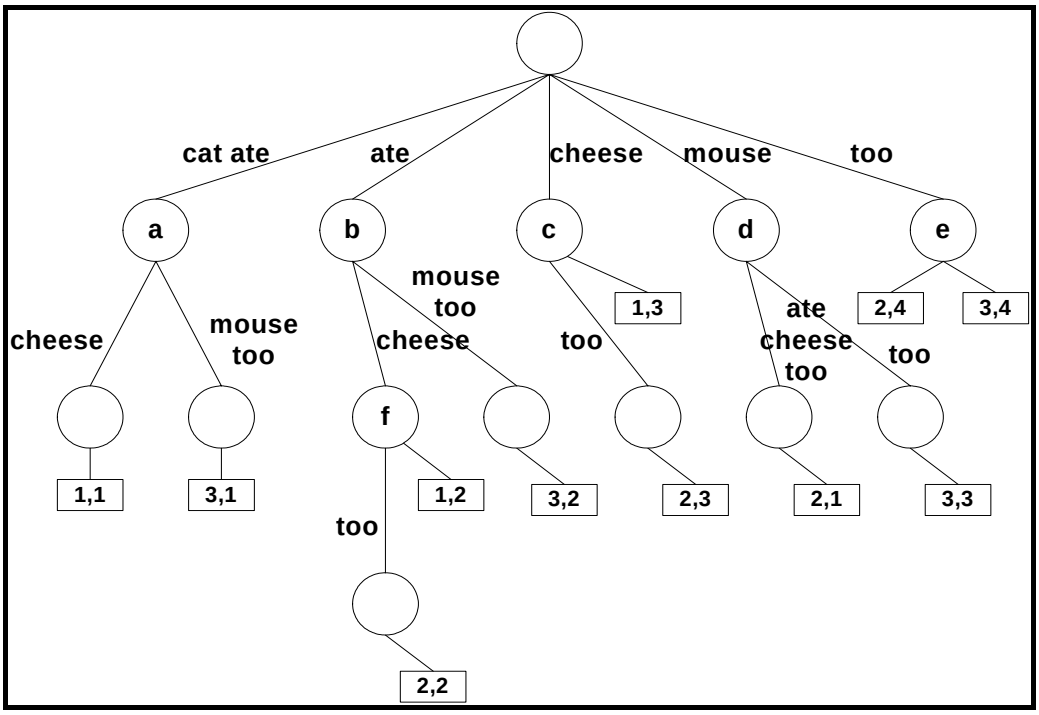


Figure 2.5: The suffix tree after compaction.

Figure 2.5 shows the final suffix tree containing a set of all suffixes of strings “cat ate cheese”, “mouse ate cheese too” and “cat ate mouse too”. All leaves nodes are drawn as circle and they have one or more attached boxes designating the string they are originated from. The first number means the suffix string number while the second means the node number of each suffix string. The internal nodes except root in the figure are labeled ***a*** through ***f*** for further reference.

Each node represents a group of documents. The label of each 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 our suffix tree. Table 2.1 lists the six marked nodes (or base clusters) from the example shown in Figure 2.5 and their corresponding phrases and documents.

Table 2.1: Six nodes (base clusters) from the example shown in Figure 2.5.

Node	Phase	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

3.) **Combining Base Clusters**

From the previous example, the same document can belong to more than one cluster. In order to identify the most important concept of the clusters, the phrase clusters are merged into larger clusters, based on the similarity defined by the common phrases of document contents.

The similarity between two clusters is measured by the amount of phrases they have in common. Thus, the similarity between cluster m_i and m_j is defined as follows:

$$sim(m_i, m_j) = \frac{c}{a + b} \qquad \textbf{(2.1)}$$

where c is number of phrases found in both m_i and m_j,
a is number of phrases in m_i,
b is number of phrases in m_j.

We define a threshold value, T1, to be a cutoff value of similarity for determining whether two clusters can be combined. That is,

if $sim(m_i, m_j) > T1$ then m_i and m_j are combined; otherwise they are not.

From the base clusters obtained in Table 2.1, we will try to combine or merge them to get larger clusters. Table 2.2 shows the similarity between each pair of base clusters measured by using **(2.1)**.

Table 2.2: The results after base clusters merging.

Base Cluster m_i			Base Cluster m_j			Similarity	Merge
Cluster	Phrase	Documents	Cluster	Phrase	Documents		
a	cat ate	1,3	b	ate	1,2,3	0.67	No
a	cat ate	1,3	c	cheese	1,2	0.33	No
a	cat ate	1,3	d	mouse	2,3	0.33	No
a	cat ate	1,3	e	too	2,3	0.33	No
a	cat ate	1,3	f	ate cheese	1,2	0.33	No
b	ate	1,2,3	c	cheese	1,2	0.67	No
b	ate	1,2,3	d	mouse	2,3	0.67	No
b	ate	1,2,3	e	too	2,3	0.67	No
b	ate	1,2,3	f	ate cheese	1,2	0.67	No
c	cheese	1,2	d	mouse	2,3	0.33	No
c	cheese	1,2	e	too	2,3	0.33	No
c	cheese	1,2	f	ate cheese	1,2	1.00	Yes
d	mouse	2,3	e	too	2,3	1.00	Yes
d	mouse	2,3	f	ate cheese	1,2	0.33	No
e	too	2,3	f	ate cheese	1,2	0.33	No

The last column of Table 2.2 suggests that the base cluster m_i and base cluster m_j should be merged or not if we set a threshold value, T1, at 0.8. After merging, four larger clusters are obtained as show in Table 2.3.

Table 2.3: The results after base clusters merging.

Node (Cluster)	Phase	Documents
a	cat ate	1,3
b	ate	1,2,3
c	ate cheese	1,2
d	mouse, too	2,3

To determine the hierarchical structure of the clusters, we will use the sub formula (SUB) defined between cluster m_i and m_j as follows.

$$sub(m_i, m_j) = \max \{ \frac{|m_i \cap m_j|}{|m_i|}, \frac{|m_i \cap m_j|}{|m_j|} \} \quad \textbf{(2.2)}$$

If $sub(m_i, m_j) >$ threshold value, T2, and $|m_i| > |m_j|$ then phrase j corresponding to cluster m_j will be a child of phrase i corresponding to cluster m_i.

If $sub(m_i, m_j) >$ T2, and $|m_i| < |m_j|$ then phrase i will be a child of phrase j.

Table 2.4 show the values of sub function between all pairs of clusters calculated using **(2.2)**.

If we set the threshold value, T2 = 0.6, the cluster hierarchy can be determined as shown in the last column of Table 2.4. Figure 2.6 shows the corresponding hierarchical structure in tree format.

Table 2.4: Subset function and cluster Hierarchy.

Cluster m_i			Cluster m_j			Sub	Child: Parent
Cluster	Phrase	Documents	Cluster	Phrase	Documents		
a	cat ate	1,3	b	ate	1,2,3	1	a : b
a	cat ate	1,3	c	ate cheese	1,2	0.5	-
a	cat ate	1,3	d	mouse, too	2,3	0.5	-
b	ate	1,2,3	c	ate cheese	1,2	1	c : b
b	ate	1,2,3	d	mouse, too	2,3	1	d : b
c	ate cheese	1,2	d	mouse, too	2,3	0.5	-

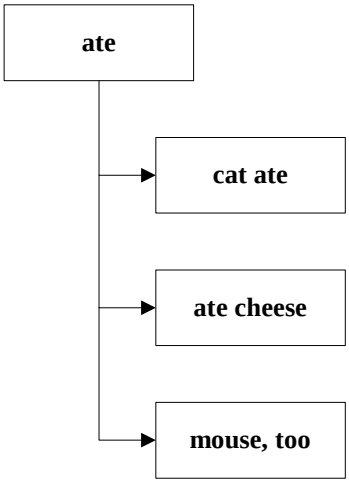


Figure 2.6: Cluster hierarchy of Table 2.4 in a tree structure.

Figure 2.7 shows a sample screen of Cluster Hierarchy results.

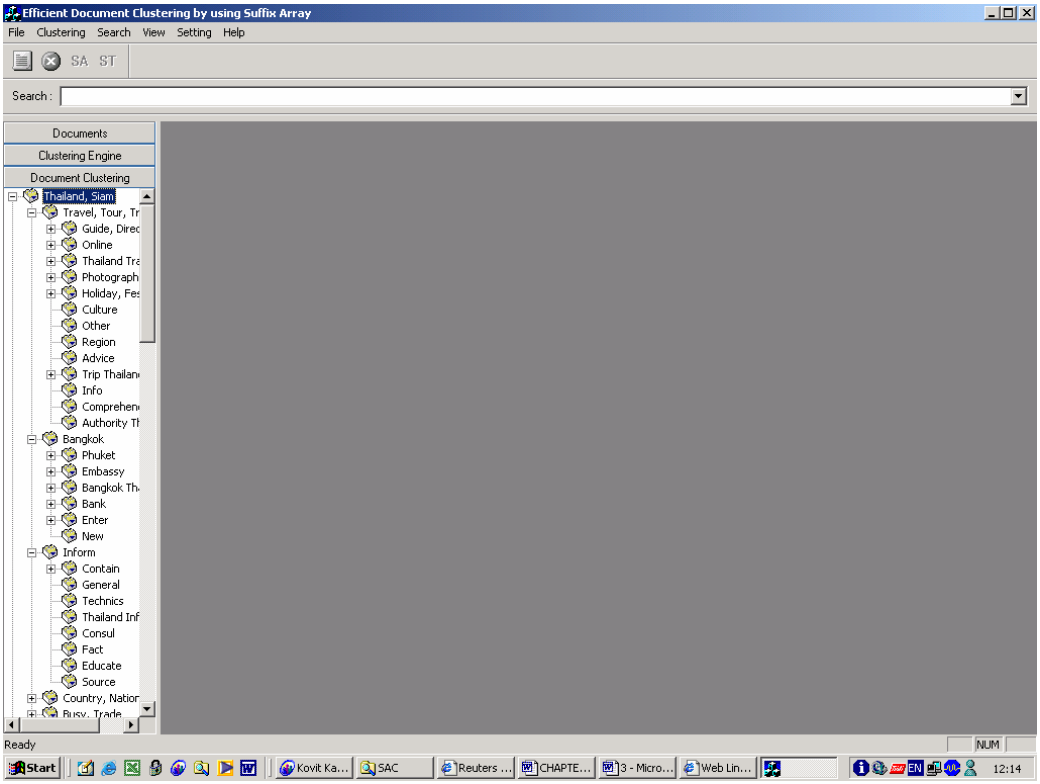


Figure 2.7: Sample clustering results in hierarchical format.

The clusters are then scored and sorted based on the scores of their base clusters and their overlapping. As the final number of clusters can be varied, only the top few clusters are reported. Typically, only the top 5-10 clusters are of interest. For each cluster, the results show the number of documents it contains, and the phrases of its base clusters.

The target of a clustering algorithm is to group each document with other sharing a common topic, but not necessarily to partition the collection. It is not reasonable 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.

In STC, documents may share more than one phrase with other documents and each document might appear in a number of base clusters. Therefore, a document can appear in more than one cluster. The overlap between clusters should not be too high; otherwise they would have been merged into a single cluster.

2.6 Single-Pass Technique

This technique will generate the results that highly dependent on a similarity threshold. To set the threshold, we should use the judgment in setting it so that we are left with a reasonable number of clusters. We review, by the example, on how Single-Pass technique work:

Suppose that we have the following set of documents and terms, that we are interested in clustering the terms using the Single-pass method. Each number represents a weight of term T_i appearing in document Doc_j.

	T1	T2	T3	T4	T5
Doc1	1	2	0	0	1
Doc2	3	1	2	3	0
Doc3	3	0	0	0	1
Doc4	2	1	0	3	0
Doc5	2	2	1	5	1

Starting from column T1 with a cluster by itself, say C1, at this point, C1 contains only one item, T1, so the centroid of C1 is simply the vector for T1 itself. That is

C1 = <1, 3, 3, 2, 2>

Next we compare (i.e., measured similarities) to the next item (T2) with centroids of all existing clusters. At this point, we have only one cluster, C1. We will use dot product as similarity for simplicity. Thus, we have

SIM(T2, C1) = 1*2 + 1*3 + 0*3 + 1*2 + 2*2 = 11.

Now we need a pre-specified similarity threshold. Let us say that our threshold is 10. This means that if the similarity of T2 and the cluster centroid is greater than 10, then we add T2 to that cluster, otherwise, we use T2 to start a new cluster.

In this case, SIM(T2, C1) = 11 > 10. Therefore we add T2 to cluster C1. We, now need to compute the new centroid for C1 (which now contains T1 and T2). The centroid which is the average vector of T1 and T2 is

C1 = <3/2, 4/2, 3/2, 3/2, 4/2>.

Now, we move to the next item, T3. Again, there is only one cluster, C1, so we only need to compute similarity between T3 and C1 centroid. The dot product of T3 and the centroid of C1 is

SIM(T3, C1) = 0 + 8/2 + 0 + 0 + 4/2 = 6.

This time, the similarity does not pass the threshold test (the similarity is less than 10). Therefore, we use T3 to start a new cluster, C2. Now we have two clusters

C1 = {T1, T2} and

C2 = {T3}.

We move to the next unclustered item, T4. Since we now have two clusters, we need to compute the similarity of T4 and the 2 cluster centroids (note that the centroid of cluster C2 right now is just the vector for T3). We have

SIM(T4, C1) = <0, 3, 0, 3, 5> . <3/2, 4/2, 3/2, 3/2, 4/2>

= 0 + 12/2 + 0 + 9/2 + 20/2 = 20.5,

SIM(T4, C2) = <0, 3, 0, 3, 5> . <0, 2, 0, 0, 1>

= 0 + 6 + 0 + 0 + 5 = 11.

Note that both similarity scores pass the threshold (10). However, we select the maximum similarity, and therefore, T4 will be added to cluster C1. Now we have the current clusters consisting of

C1 = {T1, T2, T4} and

C2 = {T3}.

The centroid for C2 is still just the vector for T3. That is

C2 = <0, 2, 0, 0, 1>,

and the new centroid for C1 is now

C1 = <3/3, 7/3, 3/3, 6/3, 9/3>.

The only item left unclustered is T5. We compute its similarity to the centroids of existing clusters as follows

SIM(T5, C1) = <1, 0, 1, 0, 1> . <3/3, 7/3, 3/3, 6/3, 9/3>

= 3/3 + 0 + 3/3 + 0 + 9/3 = 5,

SIM(T5, C2) = <1, 0, 1, 0, 1> . <0, 2, 0, 0, 1>

= 0 + 0 + 0 + 0 +1 = 1.

Neither of these similarity values pass the threshold. Therefore, T5 will have to go into a new cluster C3. There are no more unclustered items left, so clustering is done (after making a single pass through the items). The final clusters are:

C1 = {T1, T2, T4},

C2 = {T3},

C3 = {T5}.

2.7 Web Document Clustering

Many document clustering algorithms and implementation rely on the off-line clustering of the entire document collection. Since the Web search engines' collections are too large and dynamic, the off-line clustering cannot be used in this study. Therefore, the clustering has to be applied to the smaller set of documents returned in response to a query.

Considering that the search engines service millions of queries per day with free of charge, and the CPU cycles and memory dedicated to each individual query are severely curtailed. To get better performance, clustering has to be performed on a separate machine, which receives search engine results as input, creates clusters and presents them to a user. The several key requirements for Web document clustering methods have to be met as a result.

There are several key requirements that could be considered for Web document clustering methods [1]. These requirements may include:

1.) **Relevance:** Based on the nature of the documents, the method might produce clusters that manage grouping documents relevant to the user's query separately from irrelevant ones.

2.) **Speed:** The speed of execution also depends on the hardware capacity. It can be one of the key requirements which affect some users who want to filter through hundreds of documents. These users expect the clustering algorithm to compute efficiently.

3.) **Order independent:** Since the order of documents depend on an Internet search engine, the applied method should be independent of the initial order of the document.

4.) **Visualization:** The user-friendly browsing is typically required to provide the proper views of clusters, based on users' requirement.

2.8 Term Weighting

In this research, the various weighting methods [3] are mentioned in order to represent Web documents and clustering maps more clearly.

1.) **Binary weighting:** If the word does not occur in the Web document, the term weight will be zero (0). If the word occurs in the Web document, the term weight will be one (1).

2.) **Word frequency weighting:** This method gives the weight based on the number of times that the word appears in the document. For example, one word appearing in a document eight times would have four times of weight comparing with the word appearing in another document twice.

3.) **Normalization word frequency:** The normalization word frequency method could be applied to reduce the bias of word frequency weighting method when comparing the word in the longer Web pages and the word in shorter home pages.

4.) **Inverse Document Frequency (IDF) weighting:** Based on the consideration that the number of occurrences of a word should not be weighted directly to the number of occurrences of that word in another document. For example, the first occurrence of a word should be counted more than the following occurrences. The specific rule for setting weights could be used.

5.) **Normalization IDF weighting:** Similar to word frequency weighting method, normalization method could be applied for IDF weighting. The documents weighting matrix can also be weighted by normalization IDF weighting, so the longer documents are not biased for giving more weight.

2.9 Information Filtering

Even any document clustering techniques are selected, the required step is to filter out the low priority text or term to leave only the high priority one to make the more accurate in document clustering technique.

A document consists of recognizing individual terms. A term could be a significant term or non-significant term. Significant terms must identify with higher priority than less- or non- significant term. Some terms could be started from one root term and some could be equivalent to another. Moreover, in the information filtering context being considered in this thesis, hypertext documents contain more than just text. A document may consist of many fields, e.g. TITLE, HEADER, BODY, etc. Therefore, terms in different fields must be identified with different significant level.

2.9.1 Stopword Removal

The task to eliminate stopwords is to remove all of unwanted function words to prevent certain high frequency of omitted words. It can simply be done by scanning and dropping stopwords in a stop list. Francis and Kucera [4] found that the ten most frequently used words in the English language typically account for twenty to thirty percent of the terms in a document. These terms use large amounts of index storage and cause poor matches (although this is not usually a problem because of the use of multiple query terms for matching purposes).

One commonly-used approach to build a stop list is to use one of the many lists generated in the past. Francis and Kucera [4] produced a stop list of 425 words derived from the Brown corpus, and a list of 250 stopwords was published by Van Rijsbergen [4]. These lists contain many of the words that always have a high frequency, such as “a” and “ the”, and “is” but also may contain omitted words that may not have a high frequency for some text collections, such as “below”, “near”, “always”, and “that”. Note that unlike

high frequency words, omitted words do not necessarily hurt retrieval performance, and do not seriously affect storage.

Often, these words become crucial to retrieval, such as in a query "stocks with costs below X dollar", or "restaurants near the harbor".

A more suitable method of construction a stop list would be to produce a word frequency listing for the next to be indexed, and then, examine each of the high frequency words. If there is no known importance of a given word in the application, then that word can be safely placed on a stop list. An example of this procedure is the work done at National Institute of Standards and Technology (NIST) with a 25-megabyte collection of the Wall Street Journal. The top twenty-seven high-frequency words were examined, and four words were removed as possibly important ("a", "at", "from" and "to"). The remaining twenty-three words then became the stop list. The partial list of stopwords is shown in Table 2.5.

Table 2.5: Simple stopwords.

an	been	in	or	which
and	but	its	that	will
are	by	it	the	with
as	for	of	this	
be	have	on	was	

2.9.2 Stemming Algorithm

Stemming or conflation is done for two principal reasons: the decreasing in the required storage and the increasing in performance due to the use of word variants.

Many information retrieval systems also use suffixing or stemming to replace all words with their root forms. Different stemming algorithms have been used, including "standard" algorithms, and algorithms built for a specific

domain such as medical English. Three standard algorithms, an “S-Removal” stemming algorithm, the Lovins algorithm, and the Porter algorithm [2], are most often used.

The “S-Removal” stemming algorithm which is a basic algorithm for conflating singular and plural words format, is commonly used for minimal stemming. The Lovins stemmer works similarly, but on a much larger scale. It contains a list of possible suffixes, a large exception list, and many cleanup rules. In contrast, the Porter algorithm looks for about the lesser number of suffixes, producing word variant conflation intermediate between a simple singular-plural technique and Lovins algorithm. Table 2.6 shows an example of the differences among the three stemmers. The first column shows the actual words (full words). The next three columns show the words that are conflated with the original words (words that stem to the same root for that stemmer) based on three different stemmers.

Table 2.6: The differences among the three stemmmers., S-Removal, Lovins and Porter [14].

FULL WORD	S-Removal	PORTER	LOVINS
panels	panel panels	panel panels	panel panels
subjected	subjected	subjected subject subjective subjects	subjected subject subjective subjects
aerodynamic	aerodynamic aerodynamics	aerodynamic aerodynamics aerodynamically	aerodynamic aerodynamics aerodynamically aerodynamicist
heating	heating	heating heated	heating heated heat heats heater

2.10 Summary

In this chapter, we reviewed the basic concept of document clustering, background knowledge, motivation and benefits of performing document clustering. We also reviewed and surveyed many techniques of document clustering currently in use. In the next chapter, a document clustering technique using suffix array will be proposed including detail analysis and design of the algorithm for practical implementation.

CHAPTER III
SUFFIX ARRAY CLUSTERING

In this chapter, an alternative clustering technique, called Suffix Array Clustering (SAC), is proposed. SAC is similar to Suffix Tree Clustering (STC) method previously described in Chapter 2. SAC applies the array structure instead of tree structure to maintain and manipulate its document suffixes for clustering task. This chapter describes, in detail, the structure of SAC, its components, functions, algorithms and other processes necessary to perform a complete document clustering.

3.1 Structure of Suffix Array Clustering (SAC)

A structure of SAC consists of five main processes listed as follows.

1.) Lexical Analysis and Document Cleaning

2.) Creating of Suffix Array

3.) Computation of Longest Common Prefix (LCP)

4.) Determining of base clusters

5.) Combining of base clusters

Figure 3.1 shows the normal sequence of processes of SAC in performing document clustering.

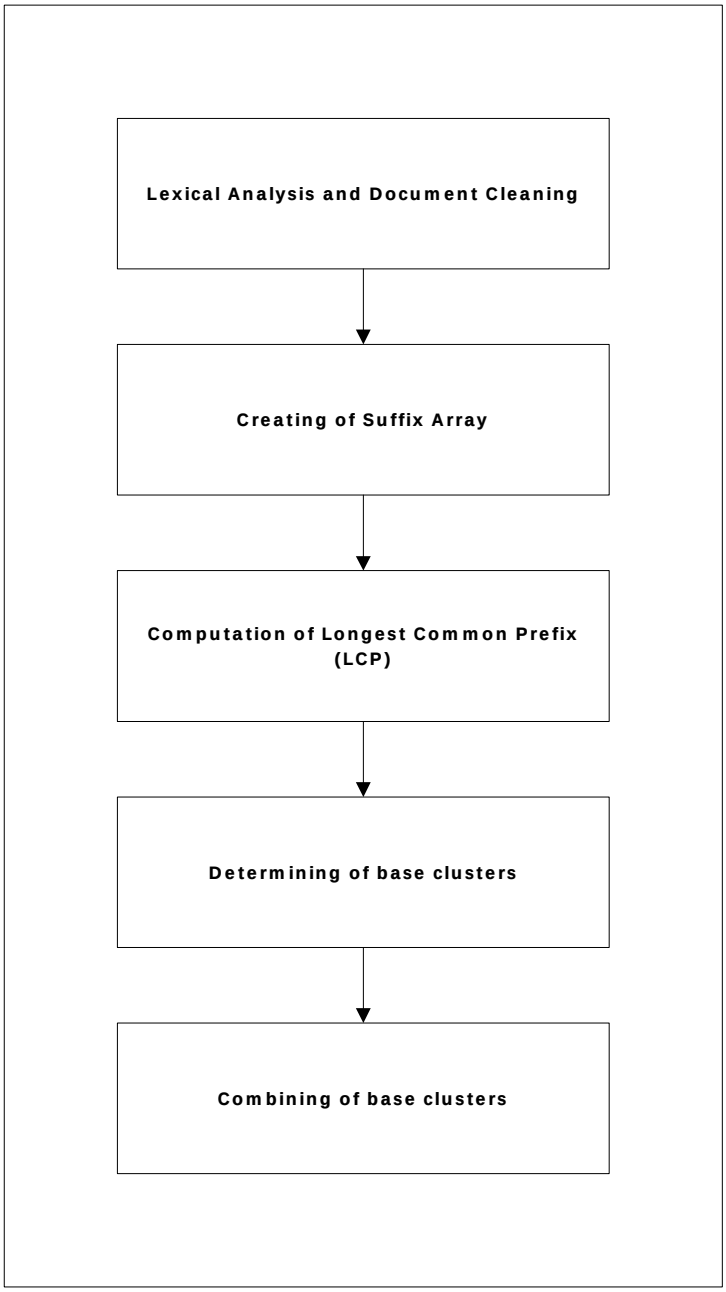


Figure 3.1: Structure of SAC.

The process of SAC is similar to STC except the data structure used in its approach. The SAC process starts with lexical analysis and document cleaning where input documents are parsed into words. Stopwords are then removed and the left are conflated using stemming algorithm.

Next, the cleaned documents are proceeded to create all suffixes and put them in an array, called suffix array. After that, LCPs are computed. Base Clusters are then

determined using suffix array and the corresponding LCPs. The final process is combining Base Clusters into larger clusters. The subsequent sections will describe in detail of each subprocess of SAC.

3.2 Lexical Analysis and Document Cleaning

This is the first process of SAC. In this step, each document will be treated as a long single string. The document will be parsed into a sequence of tokens or words. The tokens which are groups of digits will be eliminated first. The tokens containing digits or punctuation marks, such as, hyphen (-), full stop (.), underscore (_), slash (/), etc., will be determined whether they are words or not. The non-word tokens are then filtered out and the left are passed to the main process of document cleaning consisting of two subprocesses, stopword removal and stemming.

3.2.1 Stopword Removal

In this subprocess, the words having low indexing values, called stopwords, e.g. “a”, “an”, “the”, etc., will be eliminated. This can be done by matching the words against a common list of stopwords. If matching are found, then discard those words. Otherwise, we keep them for further processing. The list of common stopwords is adopted from which was derived from the Brown corpus [9]. Beside these stopwords, we add some more stopwords to our stoplist such as, “what”, “the”, “a” “an”, “for”, “to”, etc. The complete stoplist is shown in Appendix A.

3.2.2 Word Stemming (Conflation)

The next subprocess in document cleaning is stemming. Stemming is the process to relate morphologically similar words. It is used to reduce total number of different words in a document by conflating similar words together. For common English text, word conflation can be performed by table lookup, affix removal, successor variety and n-gram consideration. The Porter Stemmer [14] is one of the most popular and well known stemmers

implemented in many applications. This stemmer contains many rules in doing word conflation. However, in our SAC system, we apply a simplified version of Porter Stemmer for word stemming. The following common words can be conflated by our stemmer.

Singular - Plural nouns

Lower case – Upper case nouns

Present – Past – Past Participle verbs

3.3 Creating Suffix Array

After document cleaning was completed, a sequence of words left is parsed into all suffixes wordwise. We do not need to parse it characterwise because normally we start searching at the beginning of word. This starting point is called an indexing point. Thus, the total number of suffixes we can create from a document is equal to a total number of words left in that document after cleaning.

For examples, the documents and their suffixes created are shown as the following.

document 1 = “suwit like suwan”
suffix 1 = “suwit like suwan”
suffix 2 = “like suwan”
suffix 3 = “suwan”

document 2 = “suchai like suwan too”
suffix 1 = “suchai like suwan too”
suffix 2 = “like suwan too”
suffix 3 = “suwan too”
suffix 4 = “too”

document 3 = “suwit like suchai too”
suffix 1 = “suwit like suchai too”
suffix 2 = “like suchai too”
suffix 3 = “suchai too”
suffix 4 = “too”

We repeat this parsing for all documents. Next, all suffixes are put together and sorted alphabetically.

Table 3.1 shows a sample collection of all suffixes before sorting and Table 3.2 shows the list of suffixes after sorting. The sorted list of suffixes is then put into an array, called suffix array S and each suffix in the array is specified by S[i].

Table 3.1: Suffix array of “suwit like suwan”, “suchai like suwan too” and “suwit like suchai too” before sorting.

Suffix no.	Suffix	From document no.
1	suwit like suwan	1
2	like suwan	1
3	suwan	1
4	suchai like suwan too	2
5	like suwan too	2
6	suwan too	2
7	too	2
8	suwit like suchai too	3
9	like suchai too	3
10	suchai too	3
11	too	3

Table 3.2: The suffix array in Table 3.1 after sorting.

Suffix Array	From Suffix no.	Suffix	From document no.
S[0]	9	like suchai too	3
S[1]	2	like suwan	1
S[2]	5	like suwan too	2
S[3]	4	suchai like suwan too	2
S[4]	10	suchai too	3
S[5]	3	suwan	1
S[6]	6	suwan too	2
S[7]	8	suwit like suchai too	3
S[8]	1	suwit like suwan	1
S[9]	7	too	2
S[10]	11	too	3

3.4 Computation of Longest Common Prefix (LCP)

A Longest Common Prefix (LCP) is defined as a length (in character unit) of the longest prefix which is common to any pair of strings. A common prefix must be determined from the beginning of each prefix only. For example, the longest prefix that is common to “like suchai too” and “like suwan” is “like su”. Hence, the LCP is 7. The LCP of “suwan too” and “too” is 0. The LCP of “suchai too” and null string is 0. In our SAC system we do not need to compute LCP for all pairs of suffixes but only for the pairs where strings are located consecutively in suffix array. That is, LCP of suffix i will be determined from suffix i and suffix i-1, that is,

LCP(i) = length (max(common prefix (suffix(i), suffix(i-1)))).

The first LCP is always zero because there is no suffix prior to suffix(1). For easy implementation, we also put one additional LCP with zero value to be the last LCP. For the same set of documents previously presented in Section 3.3, all LCPs of suffixes in Table 3.2 can be computed as shown in Table 3.3.

Table 3.3: LCPs of suffix array in Table 3.2

Suffix Array	i	Suffix	LCP (i)	From Document no.
S[0]	0	like suchai too	0	3
S[1]	1	like suwan	7	1
S[2]	2	like suwan too	10	2
S[3]	3	suchai like suwan too	0	2
S[4]	4	suchai too	7	3
S[5]	5	suwan	2	1
S[6]	6	suwan too	5	2
S[7]	7	suwit like suchai too	3	3
S[8]	8	suwit like suwan	13	1
S[9]	9	too	0	2
S[10]	10	too	3	3
S[11]	11	null	0	-

However, some LPCs in Table 3.3 are not suitable for semantic clustering because they specify the common prefixes which have no meaning. For example, LCP(5) is 2. This means that S[5] = “suwan” and S[4] = “suchai too”, have the first 2 characters in common which is “su”. The common prefix “su” is meaningless and not useful for clustering. Similarly, LCP(2) is 7. The common prefix is “like su”. partial of this common prefix gives us some useful information such as “like” but some parts have no meaning such as “su”. Thus, in order to get a precise set of clusters, we should reduce each LCP to the nearest value which gives meaningful common suffix. This reduction can be done by dropping the last part of comment suffix that is not a word leaving only suffix containing complete words. We assume that complete words are separated by a space. For examples, “like su” should be reduced to “like” as LCP(2) is reduced from 7 to 4 and “su” should be reduced to null as LCP(5) is reduced from 2 to 0. Table 3.4 shows the new meaningful LCPs. These new LCPs will used for subsequent processing.

Table 3.4: Meaningful LPCs.

Suffix Array	i	Suffix	LCP (i)	Meaningful LCP (i)	From Document no.
S[0]	0	like suchai too	0	0	3
S[1]	1	like suwan	7	4	1
S[2]	2	like suwan too	10	10	2
S[3]	3	suchai like suwan too	0	0	2
S[4]	4	suchai too	7	6	3
S[5]	5	suwan	2	0	1
S[6]	6	suwan too	5	5	2
S[7]	7	suwit like suchai too	3	0	3
S[8]	8	suwit like suwan	13	10	1
S[9]	9	too	0	0	2
S[10]	10	too	3	3	3
S[11]	11	null	0	0	-

3.5 Determining of base clusters

After all LCPs were obtained, we can start clustering process by determining all base clusters. The base clusters consist of clusters that we can form by grouping documents having common suffix at certain amount not less than a given threshold value of 1, this means that documents are grouped to the same cluster if they have at least one character in common. The threshold value of 1 is normally applied to determine base clusters. This means that we try to create as many as base clusters by allowing documents to be in the same cluster if they share at least one word.

We can take the advantage of LCPs to identify base clusters conviently because LPCs inform us how many words the documents have in common with each others. The algorithm to identify base clusters in presented as follows.

Let a list of LCPs and the corresponding information are shown in Table 3.5.

Table 3.5: A general list of LCPs.

i	Suffix	LCP	From Document no.
0	S_0	l_0	i
1	S_1	l_1	j
2	S_2	l_2	k
3	S_3	l_3	l
:	:	:	:
:	:	:	:
j	S_j	l_j	x
:	:		:
:	:		:
n	S_n = null	$l_n = 0$	-

Please note that the first and the last LCPs are always zero.

From the list, we start from l_0 and put a document corresponding to l_0, document i, to the first cluster. Next, we examine the next LPC, l_1, whether it is zero. If l_1 is greater than zero we put l_0 and l_1 in the same set and put the documents corresponding to l_0 and l_1 which are document i and document j in the second cluster. Thus, we get cluster no.2 shown as follows.

Cluster no.	Document in Cluster	Common Suffix
1	i	-
2	i,j	Common Suffix 1

We continue for a larger cluster by examining the next LPC, l_2. If l_2 is greater than zero we put l_0, l_1, l_2 in the same set and put documents corresponding to l_0, l_1, l_2 which are documents i, j, k in the third cluster. Thus we get the clusters as follows.

Cluster no.	Document in Cluster	Common Suffix
1	i	-
2	i,j	Common Suffix 1
3	i,j,k	Common Suffix 2

We repeat the similar process until we find zero LPC, $l_{q.}$ Once zero LCP is found, we move the starting point from l_0 to the next one, $l_{1,}$ and then process in the similar manner. We repeat the process until the starting point is moved to $l_{q.}$ This finishes the first cycle of determining base clusters.

We continue the next cycle starting from $l_{q.}$ The process is the same as the first cycle. The cycles are repeated until the starting point is move to the last LCP, $l_{n,}$ then we stop.

To finalize the base clusters generated, we remove the smaller clusters containing a set of documents that are already contained in some other larger clusters and having the same common suffix. Figure 3.2 shows pictorially generated of the base clusters in one cycle.

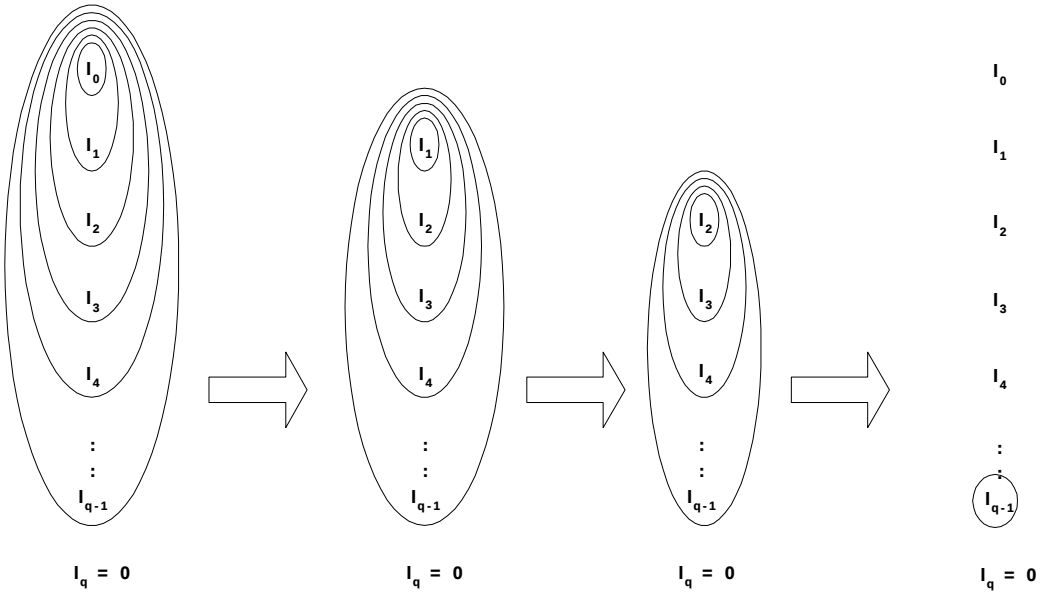
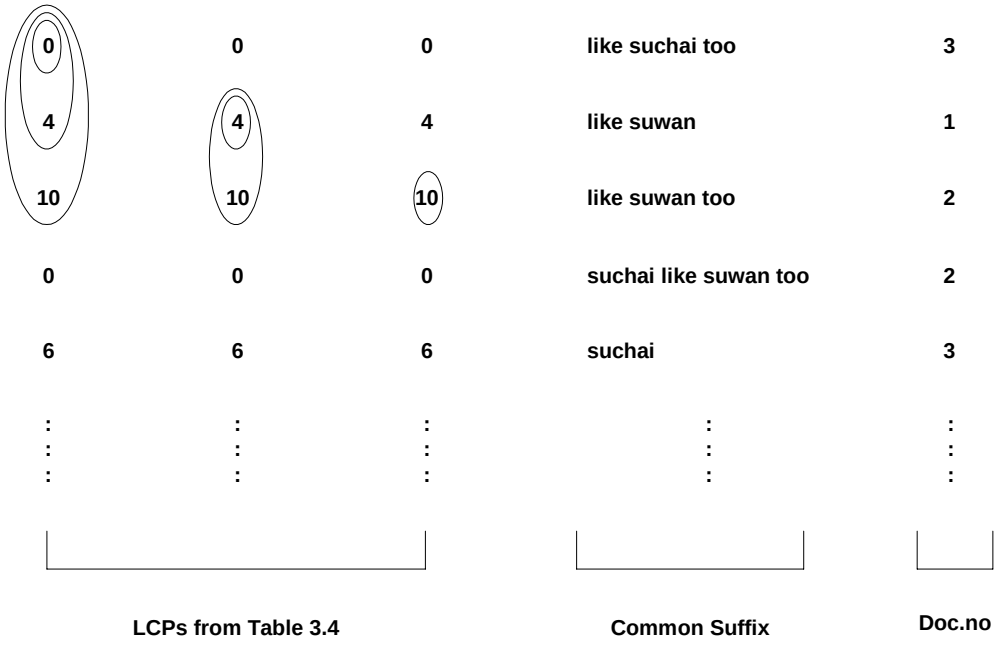


Figure 3.2: Base clustering for one cycle.

For the same example of documents and LCPs presented in Section 3.4, the base clusters are generated as follows.

First cycle:

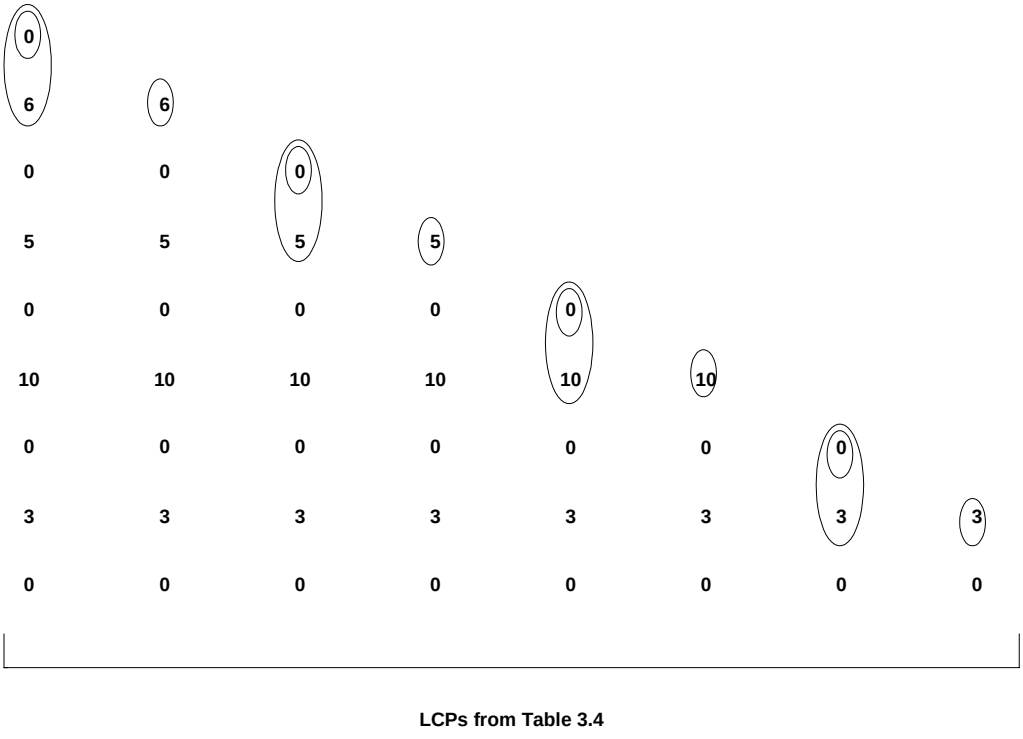


Thus, we get a set of base clusters generated in the first cluster as shown in Table 3.6.

Table 3.6: Sample base clusters for the first cycle.

Cluster no.	Document in Cluster	Common Suffix
1	3	-
2	1,3	like
3	1,2,3	like
4	1	-
5	1,2	like suwan
6	2	-

Next cycle:



For the subsequent cycles, we get more 12 base clusters. The complete base clusters are shown in Table 3.7.

Table 3.7: Complete sample base clusters.

Cluster no.	Document in Cluster	Common Suffix
1	3	-
2	1,3	like
3	1,2,3	like
4	1	-
5	1,2	like suwan
6	2	-
7	2	-
8	2,3	suchai
9	3	-
10	1	-
11	1,2	suwan
12	2	-
13	3	-
14	1,3	suwit like
15	1	-
16	2	-
17	2,3	too
18	3	-

To finalize base clusters we remove the clusters containing a single document and that document is already contained in some other larger clusters. Thus, we remove cluster 1,4,6,7,9,10,12,13,15,16 and 18. Beside these, cluster 2 can be also deleted because it contains document 1 and 3 with common suffix “like” where they are also contained in cluster 3 with the same common suffix. Please consider cluster 5 and cluster 11. They contain the same set of documents but with different common suffix. So, we cannot remove one of them from base cluster. Some others cannot be further removed because of the same reason. Finally, we get the base clusters as shown in Table 3.8.

Table 3.8: Final base clusters.

Cluster no.	Document in Cluster	Common Suffix
1	1,2,3	like
2	1,2	like suwan
3	2,3	suchai
4	1,2	suwan
5	1,3	suwit like
6	2,3	too

3.6 Combining Base Cluster

The clustering results shown in Table 3.5-3.8 and Figure 3.2 in Section 3.5 may contain too many small clusters that are still difficult for user to navigate for his/her specific interesting documents. So, it is much better if we group these clusters into larger clusters to reduce the total number of clusters. This idea can be recursively applied to make larger and larger clusters. Finally, we get a single largest cluster. This is a creation of hierarchical structure of clusters. Starting from the top level cluster, user can go deeper to get his/her required documents quickly because the hierarchical structure of cluster is already provided. The cluster hierarchy can be depicted as shown in Figure 3.3.

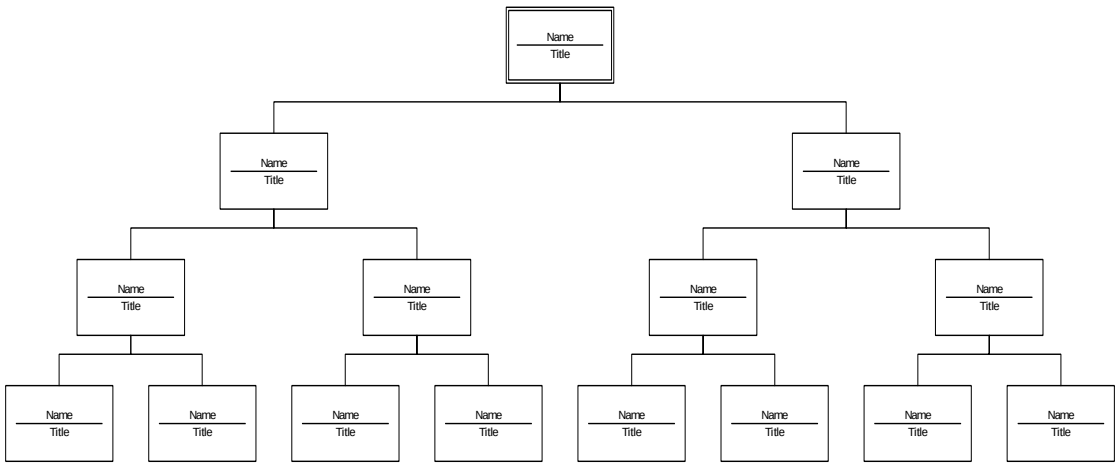


Figure 3.3: Structure of cluster hierarchy.

The process to combine the base clusters is similar to the generalization. Higher level of hierarchy we created, more general concepts we get. This results in more documents being grouped together. On the contrary, if we move down the hierarchy, more specific concepts we get, and fewer number of documents are grouped. The hierarchy will help a user discard many unwanted documents or get more related documents faster. To create such a combination (or hierarchy), it involves many tasks such as determination of similarity, creation of similarity matrix and hierarchy construction. Each task will be described in the following subsections.

3.6.1 Similarity Computation

There are many approaches to determine the similarity between two clusters, such as, the distance between centroids of the clusters, the largest or smallest distance between any document in one cluster and any document in other clusters. However, our SAC system measures the similarity between any two clusters by using the amount of documents they have in common. This is called overlap coefficient or coefficient of association [3]. Formally, normalized similarity is used and defined as follows.

$$\text{sim}(\text{cluster}_i, \text{cluster}_j) = \frac{|\ \text{documents in cluster}_i \cap \text{documents in cluster}_j\ |}{|\ \text{documents in cluster}_i \cup \text{documents in cluster}_j\ |}$$

This similarity has been already normalized. We can verify that the value of sim (cluster_i, cluster_j) never goes beyond 1 and never gives a negative value.

Example1: From the final base clusters we got in Table 3.8,

Cluster 1 consists of documents 1,2,3 with common suffix = “like”,
Cluster 2 consists of document 1,2 with common suffix = “like suwan”.

Thus $\text{sim}(\text{cluster}_1, \text{cluster}_2) = \dfrac{|\ \{1,2,3\} \cap \{1,2\}\ |}{|\ \{1,2,3\} \cup \{1,2\}\ |}$

$$= \frac{|\{1,2\}\ \ |}{|\{1,2,3\}|}$$

$$= \frac{2}{3}$$

$$= 0.67$$

Example2: From Table 3.8, similarity of cluster 2 and cluster 5 is computed as:

$$sim(cluster_2, cluster_5) = \frac{|\{1,2\} \cap \{1,3\}|}{|\{1,2\} \cup \{1,3\}|}$$

$$= \frac{|\{1\}|}{|\{1,2,3\}|}$$

$$= \frac{1}{3}$$

$$= 0.33$$

Example3: From Table 3.8, similarity of cluster 3 and cluster 6 is computed as:

$$sim(cluster_3, cluster_6) = \frac{|\{2,3\} \cap \{2,3\}|}{|\{2,3\} \cup \{2,3\}|}$$

$$= \frac{|\{2,3\}|}{|\{2,3\}|}$$

$$= \frac{2}{2}$$

$$= 1$$

Table 3.9 shows the similarity between each pair of clusters shown in Table 3.8.

Table 3.9: Similarity between each pair of clusters.

Cluster no1.	Cluster no2.	Common Suffix1	Common Suffix 2	Documents in Cluster1	Documents in Cluster2	Similarity Value
1	2	like	like suwan	1,2,3	1,2	0.67
1	3	like	suchai	1,2,3	2,3	0.67
1	4	like	suwan	1,2,3	1,2	0.67
1	5	like	suwit like	1,2,3	1,3	0.67
1	6	like	too	1,2,3	2,3	0.67
2	3	like suwan	suchai	1,2	2,3	0.33
2	4	like suwan	suwan	1,2	1,2	1.00
2	5	like suwan	suwit like	1,2	1,3	0.33
2	6	like suwan	too	1,2	2,3	0.33
3	4	suchai	suwan	2,3	1,2	0.33
3	5	suchai	suwit like	2,3	1,3	0.33
3	6	suchai	too	2,3	2,3	1.00
4	5	suwan	suwit like	1,2	1,3	0.33
4	6	suwan	too	1,2	2,3	0.33
5	6	suwit like	too	1,3	2,3	0.33

If the similarity value of $cluster_i$ and $cluster_j$ is greater than threshold 1 (T1), $cluster_i$ and $cluster_j$ will be merged together and their labels will also be merged according to the following rules.

If (common $suffix_i$ is a substring of common $suffix_j$) then

common $suffix_{ij}$ = common $suffix_j$

Else If (common $suffix_j$ is a substring of common $suffix_i$) then

common $suffix_{ij}$ = common $suffix_i$

Else

common $suffix_{ij}$ = common $suffix_i$ + common $suffix_j$

For example, if T1 is set at 0.8, only cluster pair {2,4} and cluster pair {3,6} are merged. The final results of clustering are shown in Table 3.10.

Table 3.10: The results of clusters merging for threshold = 0.8.

Cluster no.	Document in Cluster	Common Suffix
1	1,2,3	like
2	1,2	like suwan
3	1,3	suwit like
4	2,3	suchai, too

3.6.2 Hierarchy Construction

For hierarchy construction, we need to find any parent-child relationship among clusters. In SAC system, we determine the parent-child relationship by a function called, *sub*. The function will assign parent or child to $cluster_i$ and $client_j$ to display in hierarchy format, which is defined as:

$$sub(i, j) = \frac{|\{doc\ in\ cluster_i \cap doc\ in\ cluster_j\}|}{\min\{|\{doc\ in\ cluster_i\}|, |\{doc\ in\ cluster_j\}|\}}$$

If sub(i, j) is greater than threshold 2 (T2) and cluster i is greater than cluster j then common suffix associated to cluster j will be treated as a child of common suffix associated to cluster i.

If sub(i, j) is greater then T2 and cluster i is smaller than cluster j then common suffix associated to cluster i will be treated as a child of common suffix associated to cluster j.

Otherwise there is no parent-child relationship between the two clusters.

Example1: From the cluster we got in Table 3.10,

Cluster 1 consists of documents 1,2,3 with common suffix = “like” and
Cluster 2 consists of document 1,2 with common suffix = “like suwan”.

Thus $sub(cluster_1, cluster_2) = \frac{|\{1,2,3\} \cap \{1,2\}|}{\min\{|\{1,2,3\}|, |\{1,2\}|\}}$

$$= \frac{|\{1,2\}|}{|\{1,2\}|}$$

$$= \frac{2}{2}$$

$$= 1$$

If T2 is set at 0.6, then we have sub($cluster_1$, $cluster_2$) is greater then T2 and $cluster_1$ is larger then $cluter_2$. Hence, a common suffix “like suwan” will be a child of common suffix “like”.

Example2: From the clusters shown in Table 3.10, sub function of $cluster_2$ and $cluster_3$ is computed as:

$$\text{sub}(cluster_2, cluster_3) = \frac{|\{1,2\} \cap \{1,3\}|}{\min\{|\{1,2\}|, |\{1,3\}|\}}$$

$$= \frac{|\{1\}|}{|\{1,2\}|}$$

$$= \frac{1}{2}$$

$$= 0.5$$

If the same threshold value, T2, is set at 0.6, then sub($cluster_2$, $cluster_3$) is less than T2. Hence, there will be no parent-child relationship between “like suwan” of $cluster_2$ and “suwit like” of $cluster_3$.

We can determine the parent-child relationship of the other cluster pairs in a similar manner as shown in the examples. All results of calculation of the parent-child relationship of each cluster are shown in Table 3.11.

Table 3.11: Calculation of the parent-child relationship of each cluster pair.

Cluster no1.	Cluster no2.	Common Suffix1	Common Suffix 2	Documents in Cluster1	Documents in Cluster2	Relationship
1	2	like	like suwan	1,2,3	1,2	yes: “like”-“like suwan”
1	3	like	suwit like	1,2,3	1,3	yes: “like”-“suwit like”
1	4	like	suchai, too	1,2,3	2,3	yes: “like”-“suchai,too”
2	3	like suwan	suwit like	1,2	1,3	no
2	4	like suwan	suchai, too	1,2	2,3	no
3	4	suwit like	suchai, too	1,3	2,3	no

Alternatively, we can show the hierarchical structure of parent-child relationships pictorially. Figure 3.4 represents the same cluster hierarchy as shown in Table 3.11.

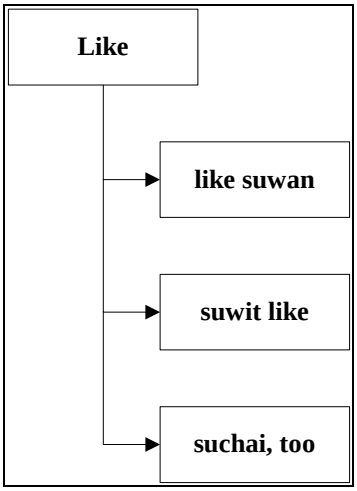


Figure 3.4: A diagram of cluster hierarchy shown in Table 3.11.

3.7 Complexity Analysis of SAC

In this section, we try to do complexity analysis of SAC. A theoretical measure of the execution of an algorithm is usually done on time or memory needed, given the problem size *n*, which is usually the number of items in a system.

One of the common complexity measurements is to use what is known as "big-oh" or "Big O" notation. We adopt this notation to our measurement of SAC. We will examine SAC part by part starting from sorting.

For sorting the suffix array in SAC, we use "Quick sort". In sorting *n* items by this algorithm, it is shown in [13] that the average case takes $O(n \log n)$ of running time and the worst case takes $O(n^2)$.

For LCP computation, it is done on a sorted suffix array. All LCPs can be computed sequentially through the array. Thus, for *n* items, LCPs can be done in $O(n)$ running time.

For determining base clusters, the complexity is depended on both the number of items, *n*, and the number of clusters, *m*. The complexity is normally $O(nm)$. However, in our SAC system, base clusters complexity is depended on number of clusters and sizes of clusters, S. For each cluster, the first cycle takes *S* running time, the second takes *S-1* running time, the third takes *S-2* running time and so on. Hence, for one cluster, it takes $S+(S-1)+(S-2)+\ldots+3+2+1 = (S(S+1)) / 2$ running time. Thus, for *m* clusters, the complexity is $O(m * (S(S+1) / 2))$ or $O(m * S^2)$, where *S* is the cluster size. For equal size clusters, we have $S = n/m$. This implies the complexity of $O(m * n^2/m^2)$ or $O(n^2/m)$. If the number of clusters is *n*, i.e., $S = 1$, then we will have the complexity of $O(n * 1^2)$ or $O(n)$.

For combining base clusters, the complexity is depended on only the number of clusters. Thus, our SAC system uses exhaustive enumeration to perform this combination. The reason that is in practical, the number of clusters is many times smaller than the number of documents such that the exhaustive enumeration works with reasonable cost. For each cluster of *m* clusters, we have to examine the other *m-1*

clusters. Hence, the running time for each cluster is *(m-1)*. This implies that the complexity is combining base clusters in $O(m(m-1))$ or $O(m^2)$.

3.8 Summary

This chapter proposed an alternative clustering technique using suffix array instead of suffix tree for more effective document clustering, called, SAC. Our method produces the same clustering precision as clustering technique using suffix tree. SAC also has complexity comparable to that of clustering technique using suffix tree. However, SAC is easier to implement and also requires less memory. In the next chapter, an implementation of SAC will be described including flow diagrams of all components, algorithms of all subroutines and functions, tools and programming techniques necessary for a prototype development.

CHAPTER IV
SYSTEM DESIGN AND IMPLEMENTATION

This chapter presents the design and implementation of Suffix Array Clustering (SAC) proposed in chapter 3. The main purpose of this chapter is to describe the design concept of SAC. We also provide the necessary information in order to understand the architecture and implementation of SAC system.

4.1 Suffix Array Clustering Architecture

At present, there are many well-known web sites available for information searching, for examples, Google, Yahoo. Unfortunately, the search results obtained from these search engines may scatter over several pages and may not be on the first page of the search results. Hence, we will use these results as our test document collection to be clustered by our SAC system. We design a prototype of SAC system in connection with Google, Yahoo and other Web sites that support searching on the Internet. Figure 4.1 shows the architecture of a Web document clustering with SAC. It consists of 4 main parts: User Interface, Internet Tool, Parser and SAC. The User Interface component accepts input query and associated parameters from user then forwards them to the Internet Tool component. The Internet Tool, then, generates URLs and submits that query to the Internet search engine and waits for the results. After getting the results back, the Internet Tool component forwards them to the Parser component for content normalization. Results from the Parser component are, then, input to SAC for document clustering. Final clustering results are then passed to the Interface component for browsing. The detailed implementation of each component is described in the following subsections.

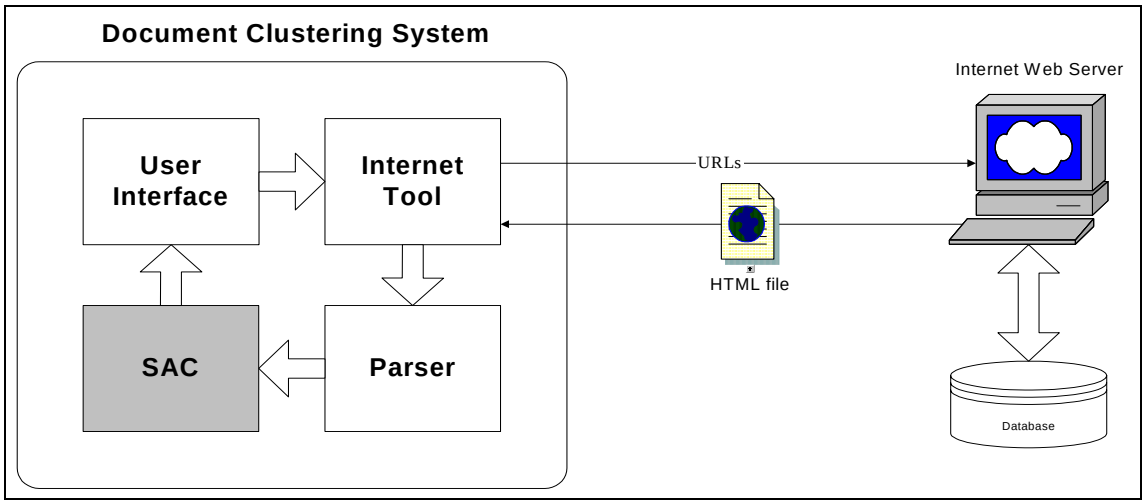


Figure 4.1: An overview of SAC implementation.

4.1.1 User Interface Component (UIC)

The User Interface is a component that provides user friendly communication between a user and the document clustering system. It accepts and passes a user query including associated parameters, e.g., number of titles, to the Internet Tool component. Figure 4.2 shows a sample screen showing query input by a user. After submitting a user query to the Internet Tool component, the User Interface component goes into a waiting mode for while. During this time, SAC system will perform its task in document clustering. When finishing, the results of clustering are returned back to the User Interface component. The User Interface, then, prepares, manages and displays such results to the output screen in hierarchical format. Figure 4.3 shows a sample of clustering hierarchy resulting from SAC.

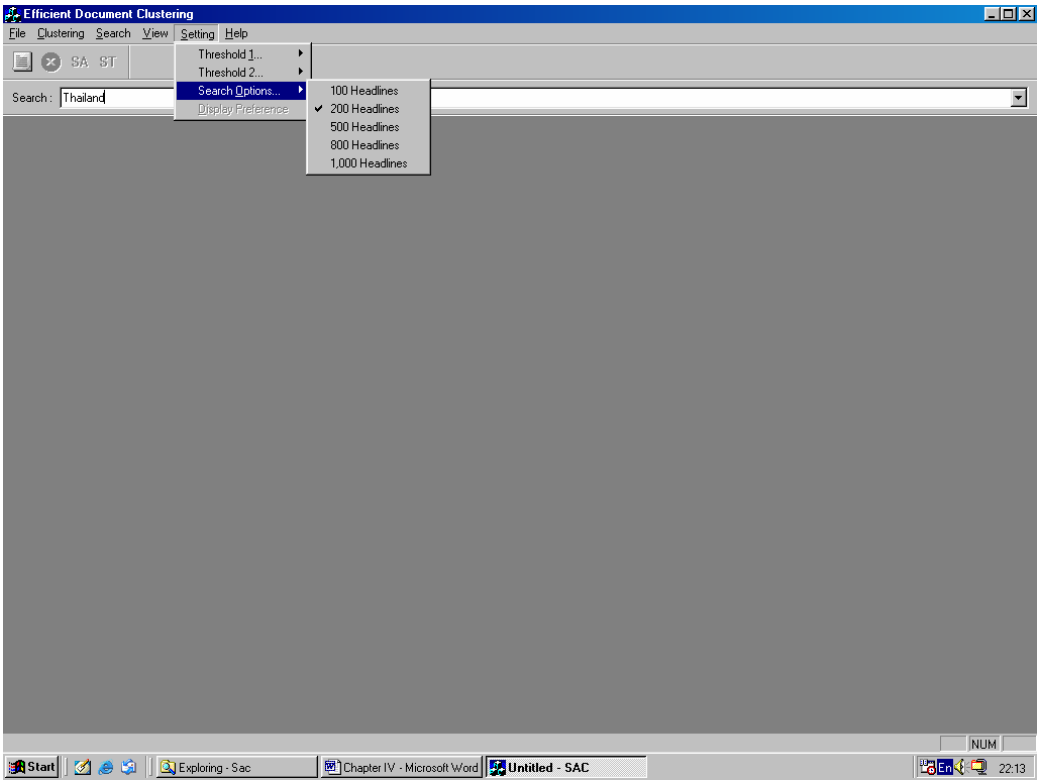


Figure 4.2: A sample input query of “Thailand” by a user.

The main task of the User Interface component is to perform I/O processing. Hence, we can use normal tools available in general purpose programming language to develop I/O dialogues directly either in text mode or graphic mode.

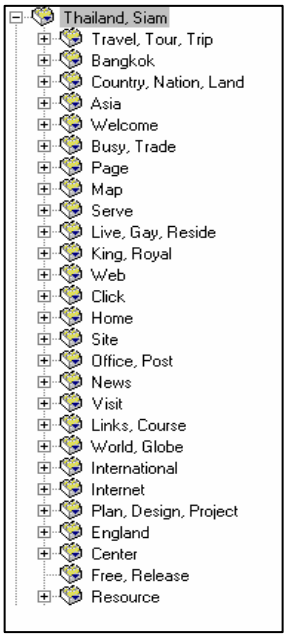


Figure 4.3: Sample output screen showing clustering results in hierarchy format.

4.1.2 Internet Tool Component (ITC)

The Internet Tool is a component in SAC system that prepares for user queries submitted to the Internet search engine and manages results returned from the Internet search engine. The Google will be the main search engine we selected for SAC testing and evaluation. The tasks of this component are started from receiving a query and a number of titles needed from a user. Next, the query and the number of titles are used to create a URL are submitted to the Internet search engine. Normally, the Internet search engines, such as Google, display the search results of 20 titles at a time. However, 20 titles of documents are not enough to evaluate the document clustering. Therefore, we do some modifications on the URL, HTTP and CGI in order to generate the URL that can be used to retrieve the search results from the Internet search engine more than 20 titles at a time.

4.1.2.1 Uniform Resource Locator (URL)

In order to get information from the Internet, each document must have an address specified by Uniform Resource Locator or URL. URLs are in the format:

protocol://computer/directory/file

A common protocol to access Web pages is HyperText Transfer Protocol (HTTP). So the URL usually begin with "http://". An example of address of a Web page is:

http://www.guinness.ie/products/draught.html

For this URL, www.guinness.ie is a domain name of a computer. An Internet Protocol (IP) addressed in numeric format may also be used. The numeric addresses are in the form **nnn.nnn.nnn.nnn**, e.g., 127.43.12.133. If there is a colon (:) and a number after the IP address, it refers to the port number, which is used if a non-standard HTTP port (i.e., a port number other than 80) is being used. For example, http://192.168.10.23:90 is an IP address using HTTP port number 90.

4.1.2.2 HyperText Transfer Protocol (HTTP)

HTTP is a HyperText Transfer Protocol. This protocol is commonly used to define communications between clients and Web server softwares.

4.1.2.3 Common Gateway Interface (CGI)

The CGI is the interface that provides communication between computer programs and HTTP or WWW servers. CGI programs typically handle information requested and return a document which is usually created dynamically, based on information contained in the

request. The term “gateway” refers to the action of converting the information requested (such as requests from SQL server) into another information useful to the client, i.e., a HTML page.

User requests will be generated in HTML format, and then passed to the program using the CGI, which uses special encoding to pass the arguments on a command line or through the URL. An argument is separated from its value by an equal sign (=), and each “argument=value” is separated from the others by an ampersand (&). All other non-alphanumeric characters (i.e. everything but a-z, A-Z, 0-9) must be preceded by a percentage sign (%) followed by a hexadecimal ASCII code of that character.

For example:

RECORD=5&SERVICE=IDN_SELECTFEED

will be encoded as:

RECORD=5&SERVICE=IDN%5FSELECTFEED

To pass arguments to a CGI process from the command line (on the server running the CGI script), the arguments must be stored in the environment variable QUERY_STRING. For example:

QUERY_STRING=RECORD=5&SERVICE=IDN%5FSELECTFEED

The arguments may also be passed to the CGI process from a client browser through the URL by placing them immediately after the CGI-program name, separated by a question mark (?).

For example, if we want to send the query about “Thailand” to the Google search engine, the URL for this query may be generated as follow.

http://www.google.com/search?q=Thailand&hl=en&num=100

where **www.google.com** is a server address,

/search is a remote path,
q= is a query keyword,
hl= is a language represented in the Google search engine,
num= is a number of titles per page.

This URL is sent to the Google website to query data by the Google search engine. Returning results will be in HTML documents and once received they are passed to the Parser component.

4.1.3 Parser Component

The Parser component is the component getting HTML documents from the Internet Tool component as its input. The Parser then removes HTML tags out of the document leaving only text. The remaining text is then passed to SAC component to perform clustering. Before starting detail implementation, we will describe some characteristics of HTML document as follows:

HTML (HyperText Markup Language) documents contain many different links, connecting user by various paths to other resources in the form of text, sound, still and moving images. Using hypertext, documents and other resources can be stored at any site in the world and linked together. Addresses, called URLs (Uniform Resource Locators), make them possible to be accessed from remote systems as well as within the local systems.

For example, a HTML result page returned from Google is shown in Figure 4.4.

```
<html>
<head>
<meta HTTP-EQUIV="content-type" CONTENT="text/html; charset=ISO-8859-1">
<title>Google Search: Thailand </title>
……
……
……
<p class=g><a href=http://www.thailand.com/ onmousedown="return clk(1,this)"><b>Thailand</b> Hotel, Travel,
Export, News, International Trade, Online <b>...</b></a><br><font size=-1><b>Thailand</b>.com offers the most
complete content about <b>Thailand</b> hotel, travel information,<br>
tours, International trade directory, guides and tools for exporting <b>...</b>
<br><font color=#008000>www.<b>thailand</b>.com/ - 7k - 21 Apr 2004 - </font><a class=fl
href=/search?q=cache:ZbOWJtvNC_wJ:www.thailand.com/+Thailand&hl=en&ie=UTF-8>Cached</a> - <a class=fl
href=/search?hl=en&lr=&ie=UTF-8&oe=ISO-8859-
1&q=related:www.thailand.com/>Similar pages</a></font>

<p class=g><a href=http://www.cia.gov/cia/publications/factbook/geos/th.html onmousedown="return
clk(2,this)">CIA - The World Factbook -- <b>Thailand</b></a><br><font size=-1> <b>...</b> Introduction,
<b>Thailand</b>, Top of Page. <b>...</b> Known as Siam until 1939, <b>Thailand</b> is the only<br>
Southeast Asian country never to have been taken over by a European power. <b>...</b>
<br><font color=#008000>www.cia.gov/cia/publications/factbook/geos/th.html - 101k - </font><a class=fl
href=/search?q=cache:fgNgAf8IIwYJ:www.cia.gov/cia/publications/factbook/geos/th.html+Thailand&hl=en&ie=UTF
-8>Cached</a> - <a class=fl href=/search?hl=en&lr=&ie=UTF-8&oe=ISO-8859-
1&q=related:www.cia.gov/cia/publications/factbook/geos/th.html>Similar pages</a></font>

<p class=g><a href=http://www.nationmultimedia.com/ onmousedown="return clk(3,this)">Welcome to The
Nation</a><br><font size=-1> <b>...</b> Travel <b>Thailand</b>. <b>...</b> Password, register now, Hotel Travel.
ป Bangkok Hotels ป Chiang<br>
Mai Hotels ป Pattaya Hotels ป Phuket Hotels ป Samui Hotels ป <b>Thailand</b> Hotels. <b>...</b>
<br><font color=#008000>www.nationmultimedia.com/ - 95k - 21 Apr 2004 - </font><a class=fl
href=/search?q=cache:BJofv49fCNgJ:www.nationmultimedia.com/+Thailand&hl=en&ie=UTF-8>Cached</a> - <a
class=fl href=/search?hl=en&lr=&ie=UTF-8&oe=ISO-8859-
1&q=related:www.nationmultimedia.com/>Similar pages</a></font>

<p class=g><a href=http://www.bangkokpost.net/ onmousedown="return clk(5,this)">Bangkok Post Main Entrance
Page</a><br><font size=-1>Thursday 22 April 2004 Last updated 8:27 AM Thai local time, more, <b>...</b>
<br><font color=#008000>www.bangkokpost.net/ - 43k - 21 Apr 2004 - </font><a class=fl
href=/search?q=cache:qFkEmCuqO5MJ:www.bangkokpost.net/+Thailand&hl=en&ie=UTF-8>Cached</a> - <a
class=fl href=/search?hl=en&lr=&ie=UTF-8&oe=ISO-8859-
1&q=related:www.bangkokpost.net/>Similar pages</a></font>
……
……
……
</body>
</html>
```

Figure 4.4: Sample documents in HTML returned from Google.

Referring to the HTML documents shown in Figure 4.4, each document or headline begins with "<p class=g> and ends with ". The Parser will remove these tags out of the HTML documents. Figure 4.5 shows the corresponding contents of HTML documents shown in Figure 4.4 but without HTML tags.

Thailand Hotel, Travel, Export, News, International Trade, Online
Thailand.com offers the most complete content about Thailand hotel, travel information,
tours, International trade directory, guides and tools for exporting

CIA - The World Factbook – Thailand
Introduction, Thailand, Top of Page. Known as Siam until 1939, Thailand is the only
Southeast Asian country never to have been taken over by a European power.

Welcome to The Nation
Travel Thailand. Password, register now, Hotel Travel. ป Bangkok Hotels ป Chiang
Mai Hotels ป Pattaya Hotels ป Phuket Hotels ป Samui Hotels ป Thailand Hotels.

Tourism Authority of Thailand.
Best view at 800 x 600 pixels | จ Copyright 2003 Tourism Authority
of Thailand

Bangkok Post Main Entrance Page
Thursday 22 April 2004 Last updated 8:27 AM Thai local time, more,...
……
……
……

Figure 4.5: Plain text after removing HTML tags.

After all HTML tags are removed, the document in plain text will be proceeded to SAC for document clustering process.

4.1.4 Suffix Array Clustering (SAC) Component

SAC is the final process of document clustering that accepts the input filtered by the Parser component. This component processes non-HTML tag documents. The process starts with stopwords elimination and stemming. All strings resulting from Stemming Algorithm are then processed for String Array to find the Longest Common Prefixes (LCP) which are used to identify Base

Clusters. We then calculate similarity values of each title in the document and combine Base Clusters. If the similarity values are higher than user-defined threshold value, then these titles will be passed to the User Interface component. Otherwise they are discarded.

4.2 Detailed Implementation

In this section, the detailed implementation of each component of the proposed document clustering, SAC, is described.

4.2.1 Interface Component (UIC) Implementation

The UIC provides convenient way of communication between user and the SAC system. UIC accepts input from user in forms of keywords and a number of resulting Web pages (or titles) required. After query processing, output are returned back to UIC and displayed to user via a computer screen. Figure 4.6 shows a flow diagram of UIC. This component can be implemented straightforwardly via I/O functions available in general programming language implemented.

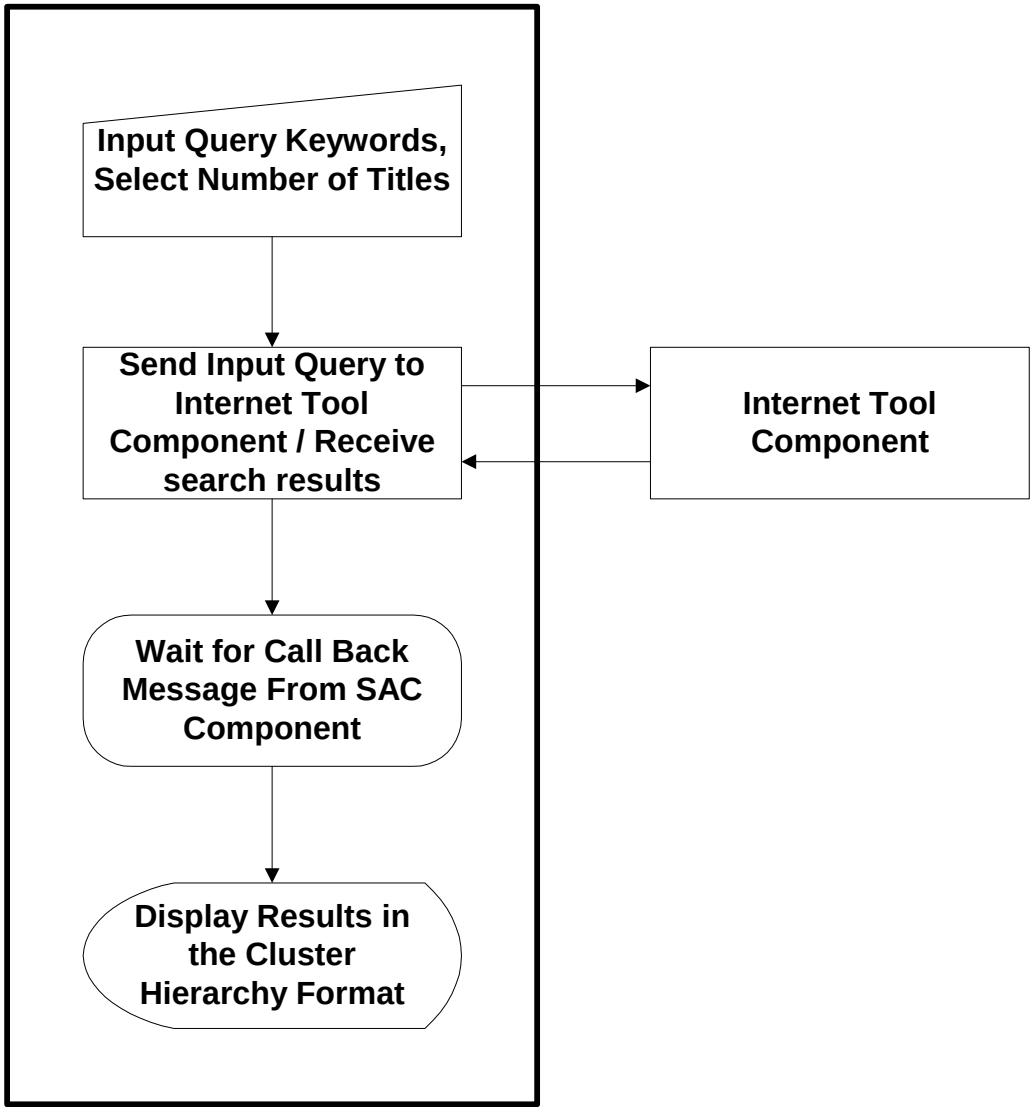


Figure 4.6: Flowchart of User Interface component.

4.2.2 Internet Tool Component (ITC) Implementation

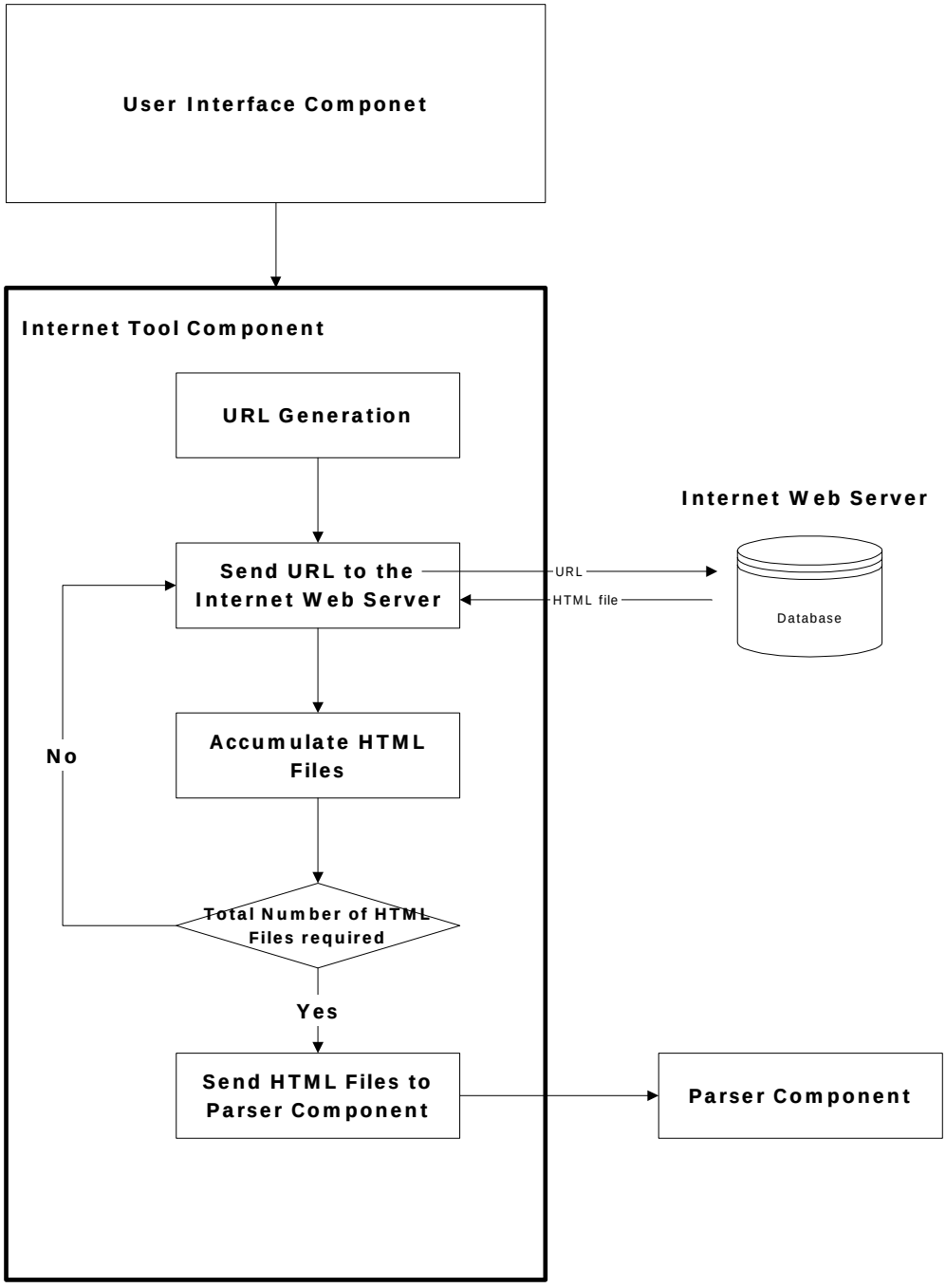


Figure 4.7: Flowchart of Internet Tool component.

As shown in Figure 4.7, ITC gets input of keywords and a number of titles required from the User Interface component. These inputs will be used to

generate URLs for searching in the Internet Web Server. The URL can be generated by concatenation of the following parameters.

1. Protocol (http is commonly used)
2. Web server name
3. Query Keywords
4. Language
5. Number of titles required to be returned

A syntax of Google URL to be created is shown as the following format.

protocal://WebServerName/search?q=keywords&hl=language&num=no.of titles

For example, if we want to search for some English documents about “Bangkok” in Google Web Server and require 50 titles of documents. The URL will be generated as the following.

http://www.google.com/search?q=bangkok&hl=en&num=50

Multiple keywords can be also used in a query by using “+” to separate each keyword such as “Computer+Education”, “Thailand+Agriculture+Weather”, etc. Table 4.1 shows more examples of URL generated for single and multiple keywords query.

Table 4.1: Examples of URL generated.

http://www.google.com/search?q=thailand&hl=en&num=50
http://www.google.com/search?q=mahidol&hl=en&num=100
http://www.google.com/search?q=sport&hl=en&num=20
http://www.google.com/search?q="document+clustering"&hl=en&num=100
http://www.google.com/search?q=thailand&hl=en&num=50
http://www.google.com/search?q=Computer+Education&hl=en&num=50
http://www.google.com/search?q=Thailand+Agriculture+Weather &hl=en&num=50
http://www.google.com/search?q=Microsoft+Price+Excel&hl=en&num=50

The Internet Web Server, after searching, returns results in HTML format to the ITC. After that, the component verifies the amount of HTML titles whether or not it is satisfied with user requirement. If not, more URL will be submitted for HTML files until the number of titles accumulate matches that of user requirement.

4.2.3 Parser Component Implementation

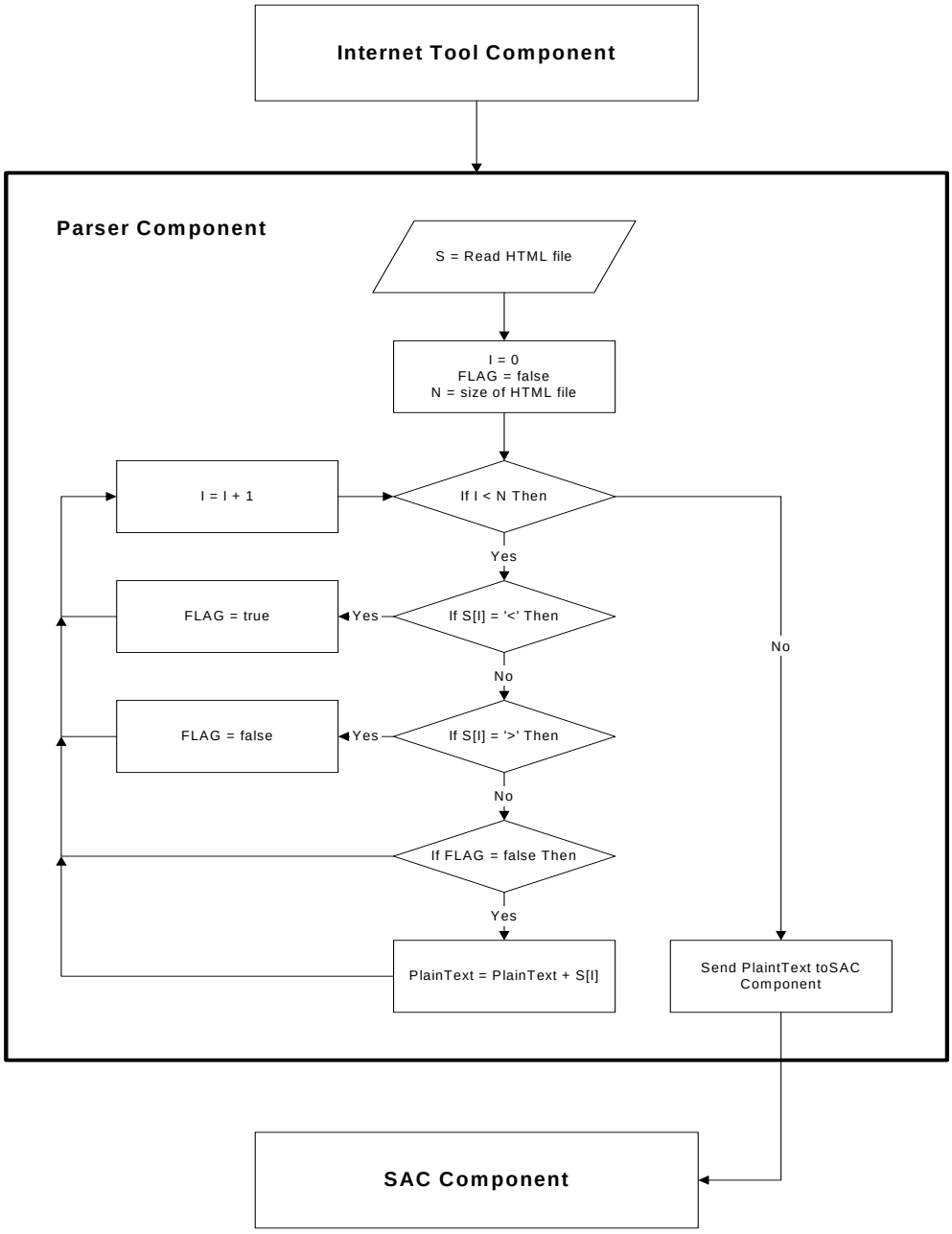


Figure 4.8: Flowchart of Parser component.

The main function of the Parser component is to convert HTML document to plaintext document by removing all HTML tags. As shown in Figure 4.8, Parser component gets HTML files resulting from the Internet Tool component and then starts the process of tag removal. HTML tags always begin with “<” and end with “>”. The contents of HTML documents are not allowed to put “<”, “>” and some other special characters directly. Hence, we can easily look for HTML tags and remove them from the documents. The flowchart in Figure 4.8 shows the steps in tag removal. The component will return only the contents in plaintext to SAC component for clustering process.

4.2.4 Suffix Array Clustering (SAC) Component Implementation

As shown in Figure 4.9, SAC is the main process of SAC system for document clustering. The first subprocess of SAC is to eliminate no-indexing value words from the document (removal of stopwords) in order to reduce number of string arrays leaving only meaningful English words. Then, these words are processed by Stemming Algorithm. All words getting from Stemming Algorithm are used to process for String Array to find the Longest Common Prefixes (LCP) which are used to identify base clusters. We then calculate similarity values between each pair of titles in documents and combine the base clusters. We use the following criteria to combine the base clusters. In the case that similarity value is higher than user-defined threshold value, then, these titles are put in the same cluster. The process is repeated until we get unchange clusters. The results from this component are then delivered to the User Interface component in cluster hierarchy format for displaying. The following subsections describe, in pseudo codes, each SAC subprocess.

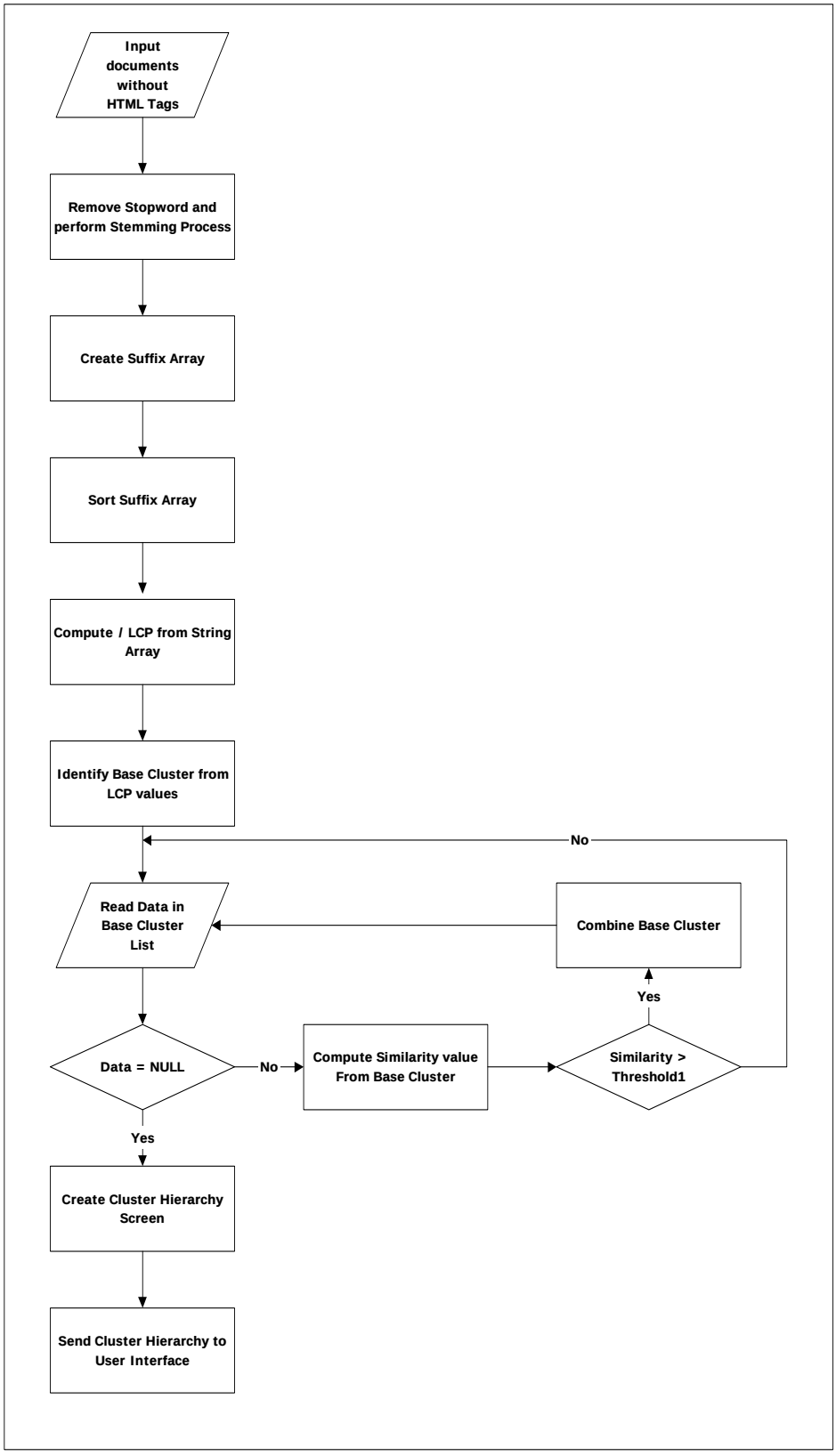


Figure 4.9: Flowchart of SAC component.

4.2.4.1 Stopword Elimination and Stemming Process

This subprocess consists of two tasks: stopword elimination and stemming process. Firstly, we do stopword removal by matching all words extracted from documents against the list of all given stopwords. If matching is found, we will remove these words from the documents. Otherwise, we will leave them as usual. A complete list of common stopwords is shown in Appendix A. Secondly, we perform stemming process. For this process, we use a simplified version of Porter stemming [14]. The process is limited to the term conflation of plural-singular nouns, present-past-past participle verbs for the normal case. Figure 4.10 shows the algorithms of stopword removal and stemming in pseudo-code format.

```
Procedure Stopword(W)

1. [Initialize]
        LW <- array of stopwords (LW is list of all given stopwords)
2. [Remove Stopword]
        For I=1 to array size of LW
                If (LW[I] = W) then
                        Return (W is stopword)
                Endif
        Next
3. [Stemming Process]
        If ends of W in “ies” then
                “ies” -> y
        Else If ends of W in “ed” then
                “ed” -> “e”
        Else if ends of W in “s”
                “s” -> NULL
        Else if ends of W in “es”
                “es” -> NULL
        End if
        PlainText = PlainText + W

4. [Finished]
        Return
```

Figure 4.10: Pseudocode of stopword elimination and stemming process.

4.2.4.2 Creating Suffix Array

After document cleaning process was completed, a sequence of words left is parsed into all suffixes wordwise. We do not need to parse them characterwise because normally we start searching at the beginning of word. This starting point is called an indexing point. Thus, the total number of suffixes we can create from a document is equal to a total number of words left in that document after cleaning. Suffix array will be associated with string and document identification which are kept for later references. Figure 4.11 shows a pseudocode for creating Suffix Array.

```
Procedure Creating of Suffix Array

1. [Initialize]
        LP <- create array of struct
                struct
                {
                        plaintext <- string
                        id <- integer
                        lcp <- integer
                }LP

        N <- number of document
        S1 <- string
        S2 <- string
        P <- integer

2. [Perform Creating of Suffix Array]
        For I=0 to N-1
                S1 = Read document from file(I)
                S2 = S1
                LP.plaintext = S2       \\ add document source into array
                LP.id = I               \\ add document id. into array
                LP.lcp = 0              \\ add default value of LCP
                Add_Array(LP)

                While (TRUE)
                        P = Find (S1, ' ')      \\ searches space for the match of a substring.
                        If (P == -1)
                                Break
                        S2 = Mid(S1, P)

                        LP.plaintext = S2  \\ add document source into array
                        LP.id = I  \\ add document id. into array
                        LP.lcp = 0  \\ add default value of LCP
                        Add_Array(LP)
                Do
        Next

3. [Finished]
        Return
```

Figure 4.11: Pseudocode for creating suffix array.

4.2.4.3 Sorting Suffix Array

This process is devised to sort a list of suffixes in alphabetical order. The Quick Sort algorithm [13] is applied for this process. A sorted list of strings is required for LCP calculation which will be used in determining of base clusters. Figure 4.12 shows the implementation of Quick Sort for Suffix Array Sorting.

```
Procedure QUICK_SORT(LP, LB, UB) ; LP is array of struct

1. [Initialize]
        FLAG <- true

2. [Perform Sorting]
        If LB < UB then
                I <- LB
                J <- UB + 1
                KEY <- LP[LB]
                Repeat while FLAG
                        I <- I + 1
                        Repeat while LP[I] < KEY (scan the keys from left to right)
                                I <- I + 1
                        Repeat while LP[J] > KEY (scan the keys from right to left)
                                J <- J - 1
                        If I < J then
                                LP[I] <---> LP[J] (interchange records)
                        else
                                FLAG <- fales
                End while
                LP[LB] <---> LP[J] (interchange records)
                Call QUICK_SORT(LP, LB, J - 1) (sort first subtable)
                Call QUICK_SORT(LP, J + 1, UB) (sort second subtable)
        End if

3. [Finished]
        Return
```

Figure 4.12: Pseudocode of sorting process.

4.2.4.4 Computation of Longest Common Prefix (LCP)

The Longest Common Prefix (LCP) values are required in the process of identifying base cluster. LCP can be computed by finding the length of longest prefix common to any pair of strings. In SAC system, LCPs are computed for any pair of strings consecutively located in the sorted list of suffixes. The pair of two strings consists of a current suffix and its preceding suffix in the sorted suffix array. In another words, LCP(i) is the length of longest common prefix between Suffix(i-1) and Suffix(i), that is,

LCP(i) = length(max(common(prefix(i-1), prefix(i))))

Figure 4.13 describes, in pseudocode, the implementation of LCP computation.

```
1. [Initialize]
        LP <- array of struct
        (Example : LP.plaintext[1] = “cat ate cheese”, LP.plaintext[2] = “ate cheese”)

2. [Computation of LCP]
         For I=1 to (array size of LP) – 1
                For J=0 to string length of LP[I] - 1
                        If (LP[I-1][J] <> LP[I][J]) then
                                Break
                        End if
                Next
                LP.lcp[I] = j
        Next

3. [Finished]
        Return
```

Figure 4.13: Pseudocode of computation of LCP.

4.2.4.5 Identifying Base Clusters

A base cluster is the cluster that is initially identified from a set of documents at the beginning of clustering process. A set of base clusters can be identified by using LCPs previously computed in Section 4.2.4.3. We combine any pair of documents together to form a base cluster whenever the LCP between two phrases contained in those documents is greater then zero and the common prefix contain the whole word, i.e., the common prefix ends with a word break character, e.g. space, period, comma, semi-colon, question mark, etc. Figure 4.14 shows the algorithm, in pseudocode, to identify the base clusters.

```
1. [Initialize]
        LP <- array of struct
        BC <- create array of struct

                struct
                {
                        text <- string
                        id <- list of document id.
                }BC

2. [Identify Base Cluster]
        For I=1 to (array size of LP)
                If (LP.lcp[I] > 0) and (LP.plaintext[I][LP.lcp[I]] = space or LP.plaintext[I][LP.lcp[I]]
= NULL) then
                        BC.text = LP.plaintext[I][LP.lcp[I]])
                        Add_List (BC.id, I)
                        Add_List (BC.id, I-1)

                        Add_Array(BC)
                End if
        Next

3. [Finished]
        Return
```

Figure 4.14: Pseudocode for identifying the based clusters.

4.2.4.6 Computation of Cluster Similarity

The similarity of base clusters in SAC system is computed from the amount of documents they are in common. Hence, similarity is computed as the ratio of the size of intersection and union sets of documents contained in both clusters. That is

$$\text{sim}(\text{cluster}_i, \text{cluster}_j) = \frac{|\text{documents in cluster}_i \cap \text{documents in cluster}_j|}{|\text{documents in cluster}_i \cup \text{documents in cluster}_j|}$$

For example, given 2 cluster *a* and cluster *b* contain sets of documents as follows.

cluster a = {doc1, doc3, doc5}
cluster b = {doc3, doc4, doc5, doc6}

We have,

$$\text{sim}(\text{cluster } a, \text{cluster } b) = \frac{|\{\text{doc3, doc5}\}|}{|\{\text{doc1, doc3, doc4, doc5, doc6}\}|} = \frac{2}{5} = 0.4$$

Amount of similarity value computed is then tested against threshold value, T1, to determine whether both clusters are similar or not. Figure 4.15 shows a pseudocode for computation of cluster similarity.

```
1. [Initialize]
        BC <- array of struct
        CBC <- create array of struct

                struct
                {
                        text <- string
                        id <- list of document id.
                }CBC

        SIM <-  float
        T1 <- float     // T1 = 0.8

2. [Computation of Cluster Similarity]
        For I=0 to (array size of BC) - 1
                For J=1 to (array size of BC)
                        SIM = Intersec (BC.id[I], BC.id[J])) / Union (BC.id[I], BC.id[J]))
                        If (SIM > T1) then
                                :
                            //  Both cluster are similar and one cluster should be included in another cluster.
                                :
                        End if
                Next
        Next

3. [Finished]
        Return
```

Figure 4.15: Pseudocode for computation of cluster similarity.

4.2.4.7 Combining Base Clusters

The combining base clusters are determined using the similarity between two clusters defined as follows.

$$sub(cluster_i, cluster_j) = \max\left(\frac{|\text{docs in } cluster_i \cap \text{docs in } cluster_j|}{|\text{docs in } cluster_i|}, \frac{|\text{docs in } cluster_i \cap \text{docs in } cluster_j|}{|\text{docs in } cluster_i|}\right)$$

To prevent fault grouping, we need a threshold value (T2) for cut off similarity. In our experiments on SAC system we set T2 to 0.6. This means that if similarity between any two clusters is greater than 0.6 then they are combined into one larger cluster. Otherwise, they are left as separated clusters. Further more, if cluster *i* and cluster *j* are combined and cluster *i* is larger than cluster *j*, a phrase corresponding to cluster *j* will be treated as a child of a phrase corresponding to cluster *i*. Otherwise, a phrase corresponding to cluster *i* will be treated as a child of a phrase corresponding to cluster *j*. Figure 4.16 shows a pseudocode for combining base clusters.

```
1. [Initialize]
        CBC <- array of struct
        SUB1 <-  float
        SUB2 <-  float
        T2 <- float     // T1 = 0.6
2. [Computation of Cluster Similarity]
        For I=0 to (array size of CBC) - 1
                For J=1 to (array size of CBC)
                        SUB1 = Intesec (CBC.id[I], CBC.id[J]) / (list size of CBC.id[I])
                        SUB2 = Intesec (CBC.id[I], CBC.id[J]) / (list size of CBC.id[J])
                        If (SUB1 > T2) or (SUB2 > T2) then
                                If (list size of CBC.id[I] > list size of CBC.id[J]) then
                                // CBC.text[J] will be treated as the child of CBC.text[I]
                                Else
                                // CBC.text[I] will be treated as the child of CBC.text[J]
                                End if
                        End if
                Next
        Next
3. [Finished]
        Return
```

Figure 4.16: Pseudocode for combining of base clusters.

4.3 Sample Implementation of SAC

This section presents a sample implementation of SAC system. We capture some I/O screens when running SAC in an actual situation. The Google Web site is the main Internet search engine used to provide the input Web documents for SAC system. Figure 4.17 shows the input screen for user to enter his/her query.

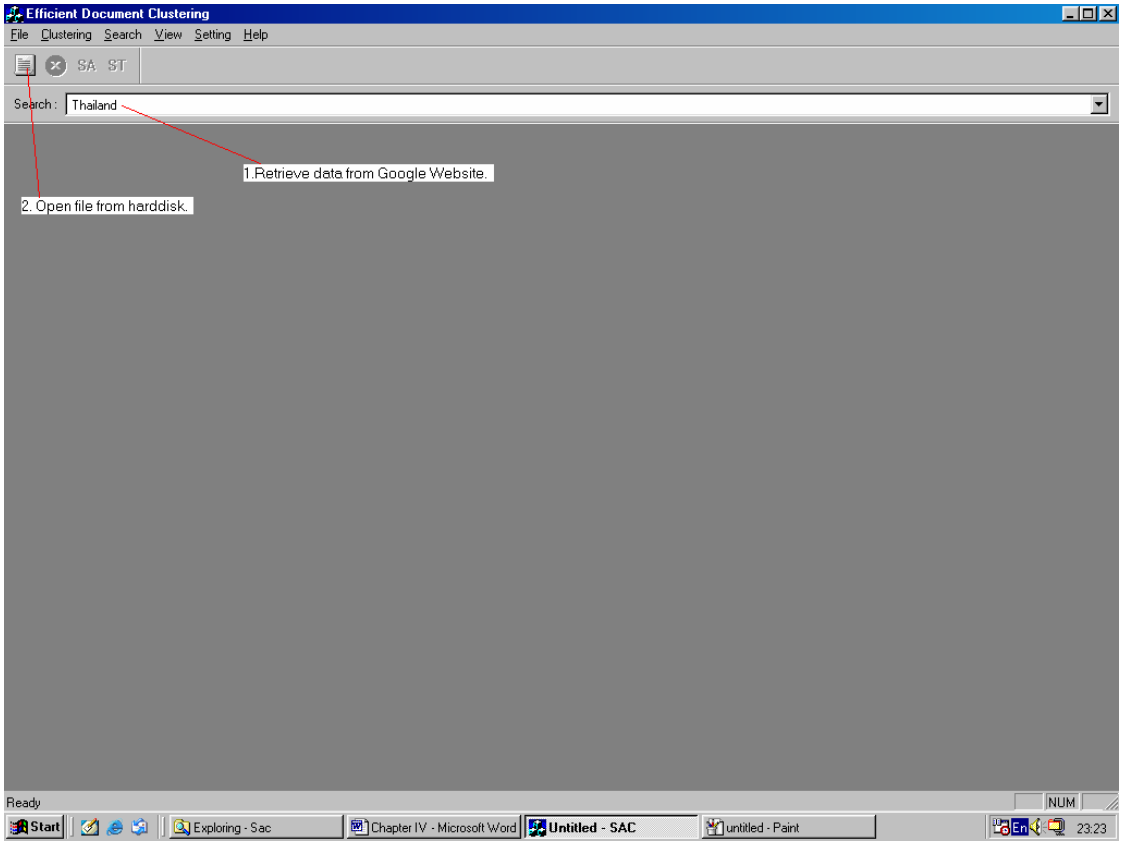


Figure 4.17: A sample input screen of SAC program.

When retrieving documents from Google Web site, the number of headlines or titles of Web documents that required to be returned can be set by "Menu Setting→Search Options" as shown in Figure 4.18.

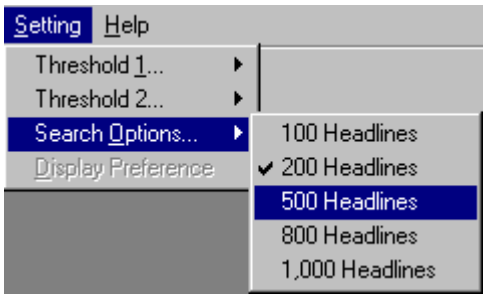


Figure 4.18: Menu to select number of headlines.

After user entered a query keyword, for example “Thailand”, together with number 500 specifying amount of titles required and submitted to the Web server, the first 500 headlines (titles) found are returned as shown in Figure 4.19.

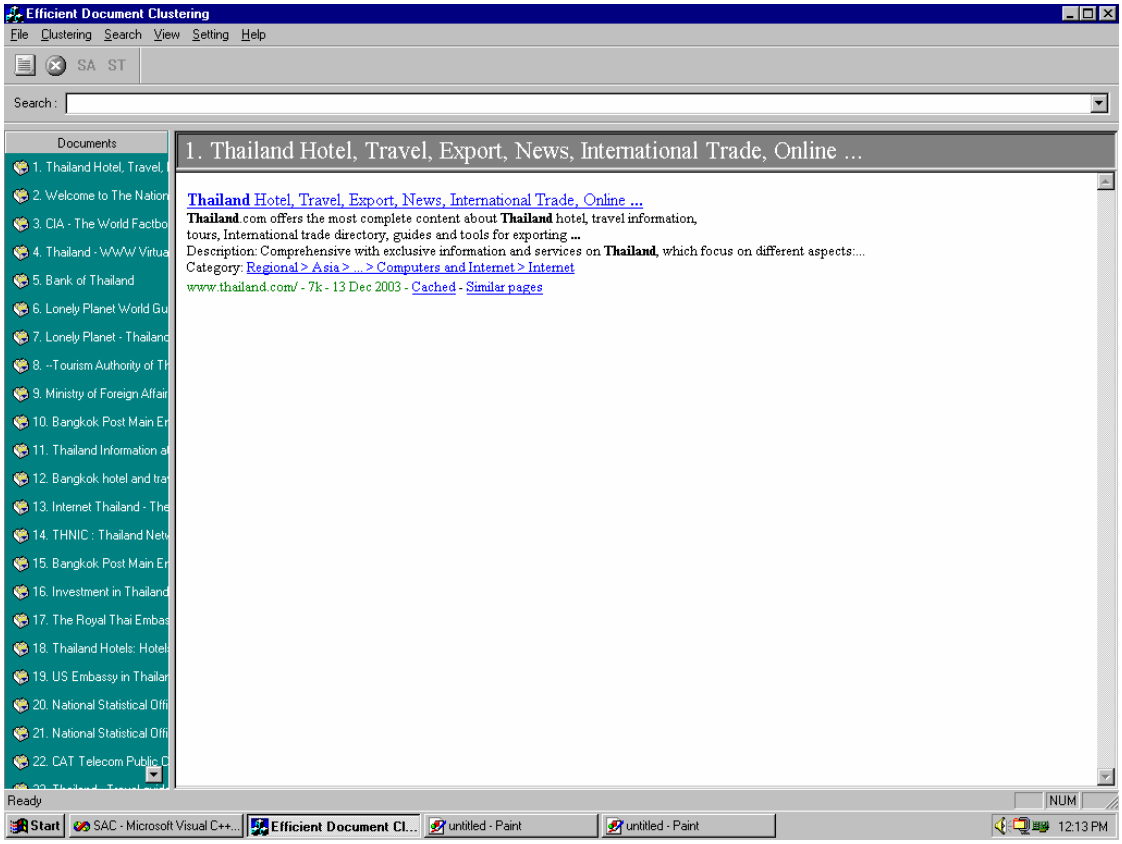


Figure 4.19: The results of searching for query “Thailand”.

Before processing the document clustering, the threshold value must be set for calculation the similarity value of each document by setting “Menu

Setting→Threshold1 or Threshold2" as shown in Figure 4.20. *Threshold1* and *Threshold2* indicate cutoff similarity values of 0.8 and 0.6 respectively. After all required parameters have been set, document clustering can be initiated by selecting a menu "clustering" as shown in Figure 4.21.

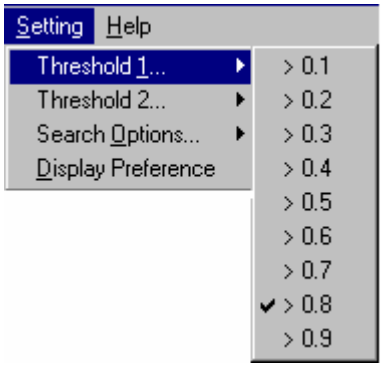


Figure 4.20: Menu to set the threshold values.

Figure 4.21: Menu to select method of document clustering.

After document clustering has been completed, the results are displayed on the screen as shown in Figure 4.22 to 4.26.

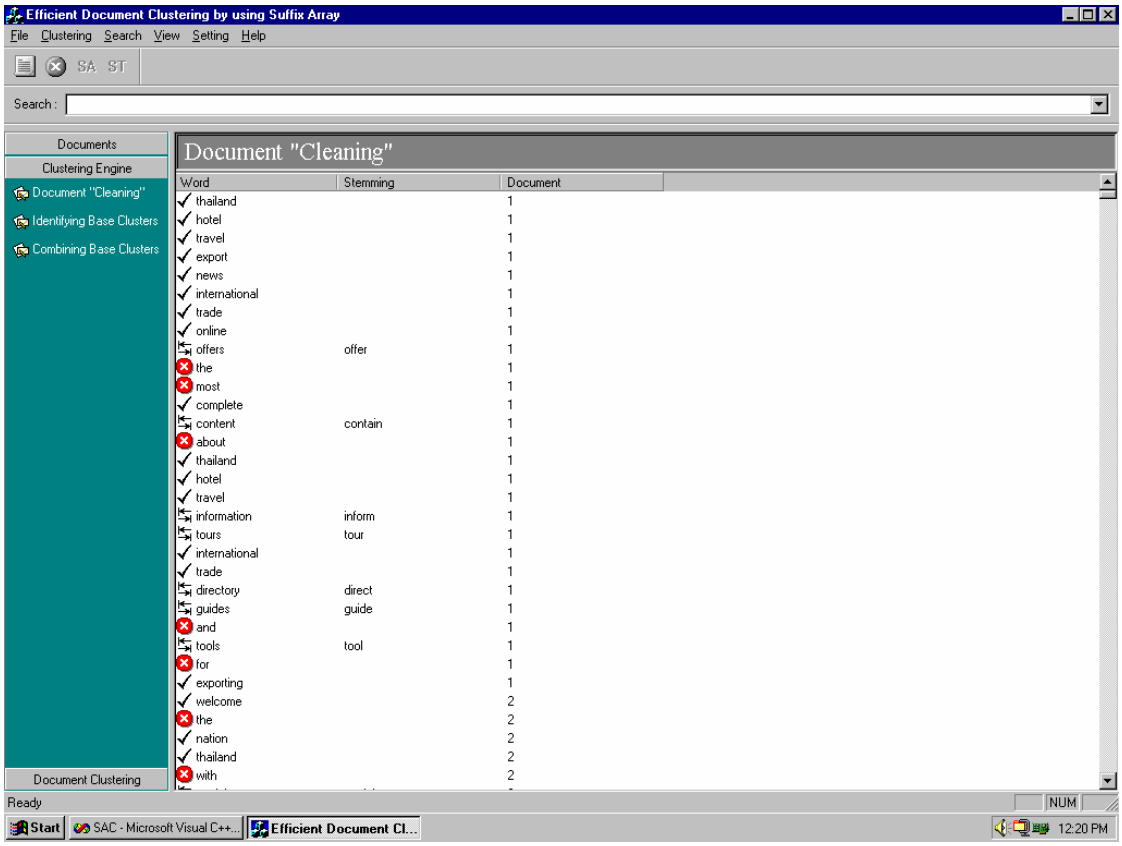


Figure 4.22: Screen of stopwords (marked with ⊗) and stemmed words (marked with ⇤⇥).

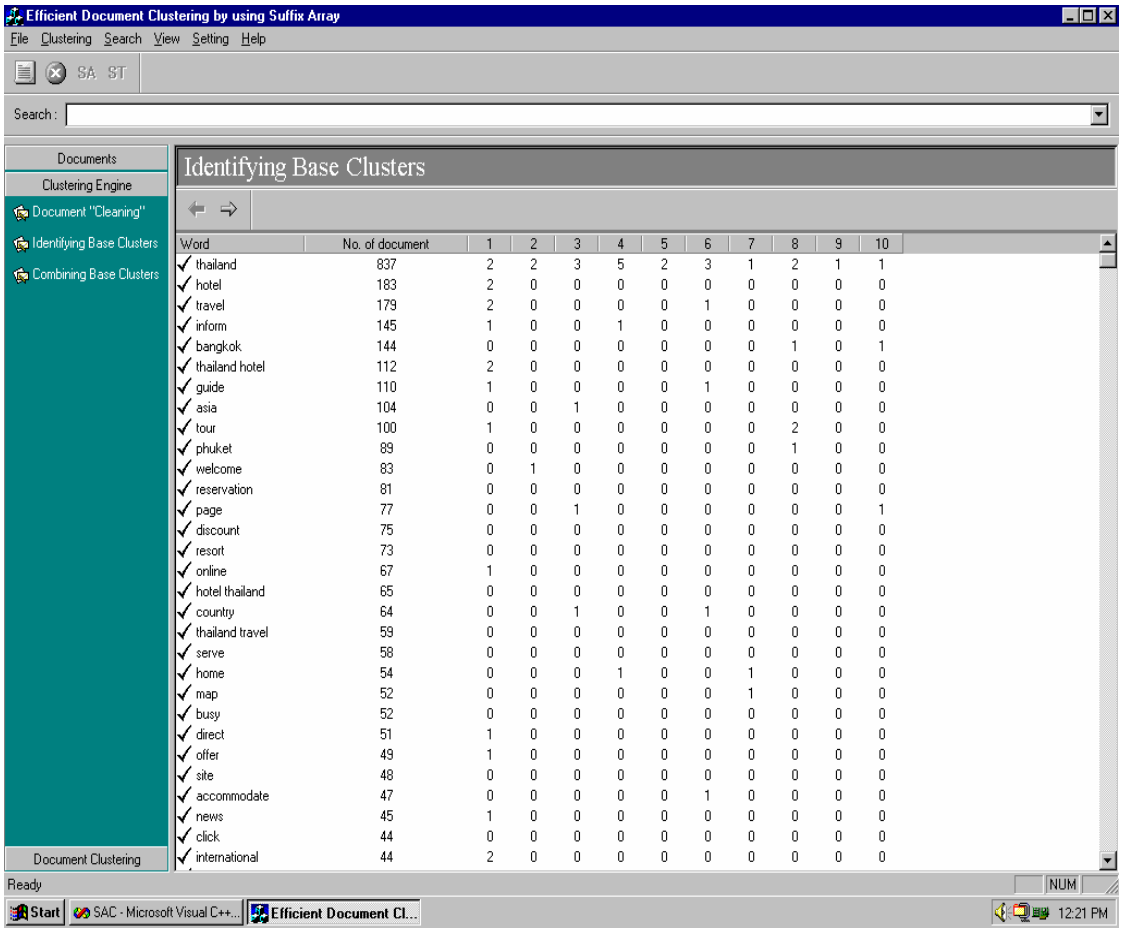


Figure 4.23: Screen displaying identified base clusters.

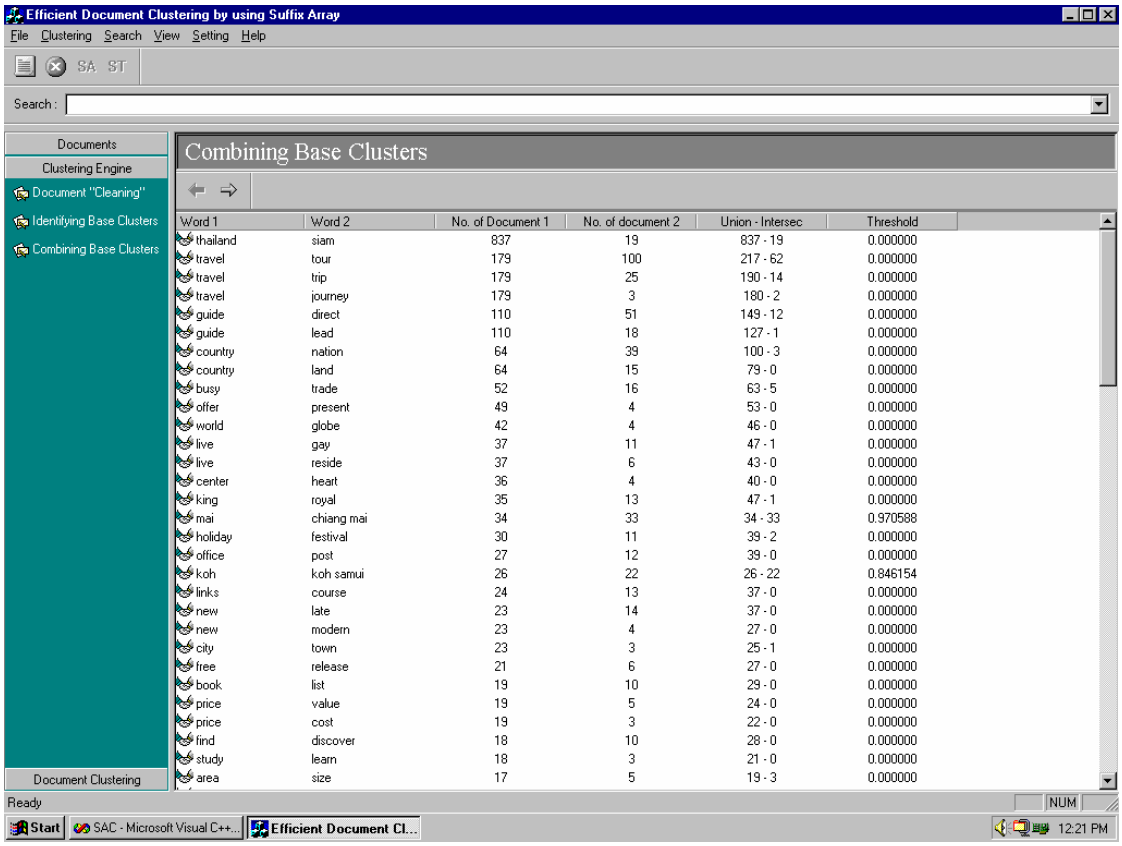


Figure 4.24: Screen of combining base clusters.

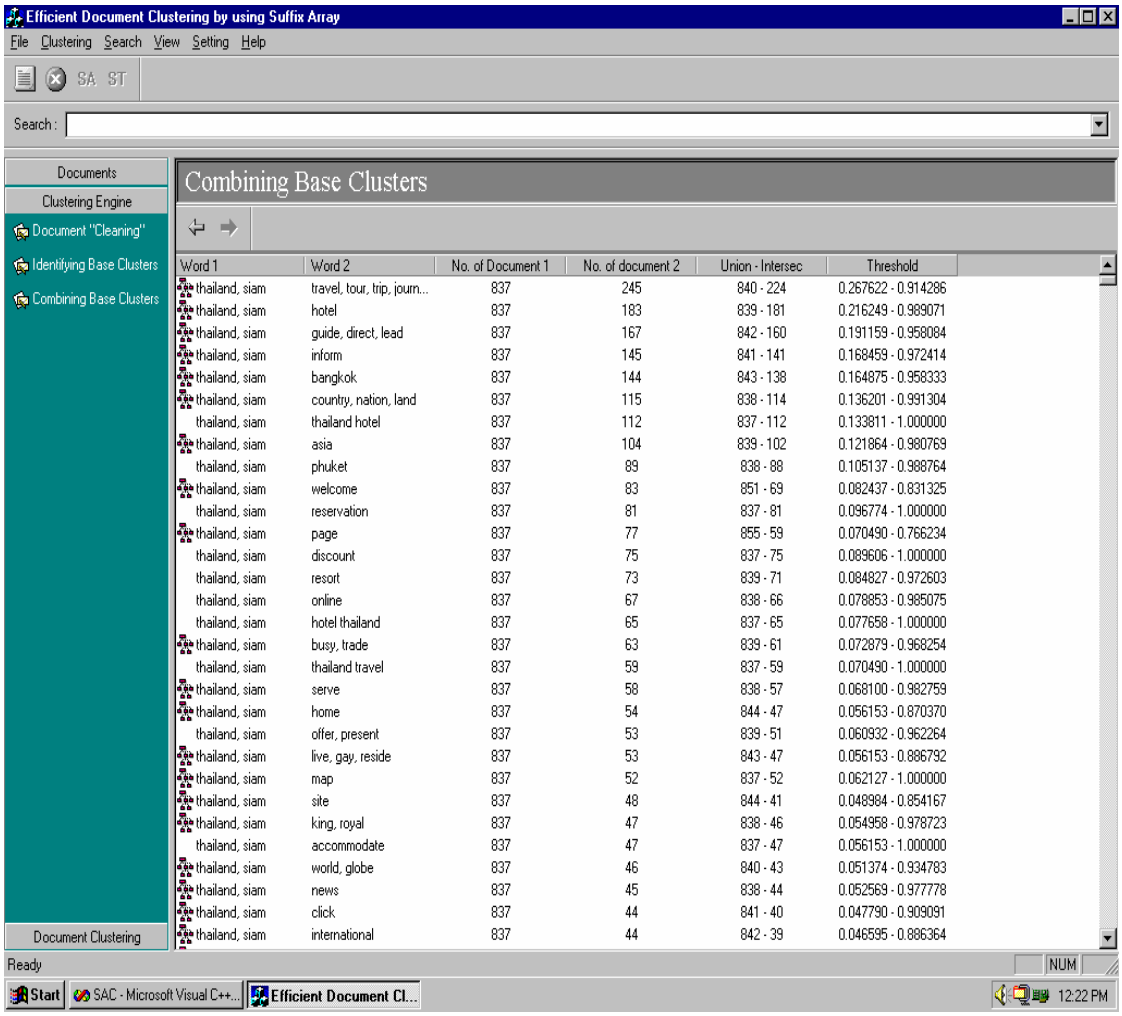


Figure 4.25: Screen displaying combined base clusters.

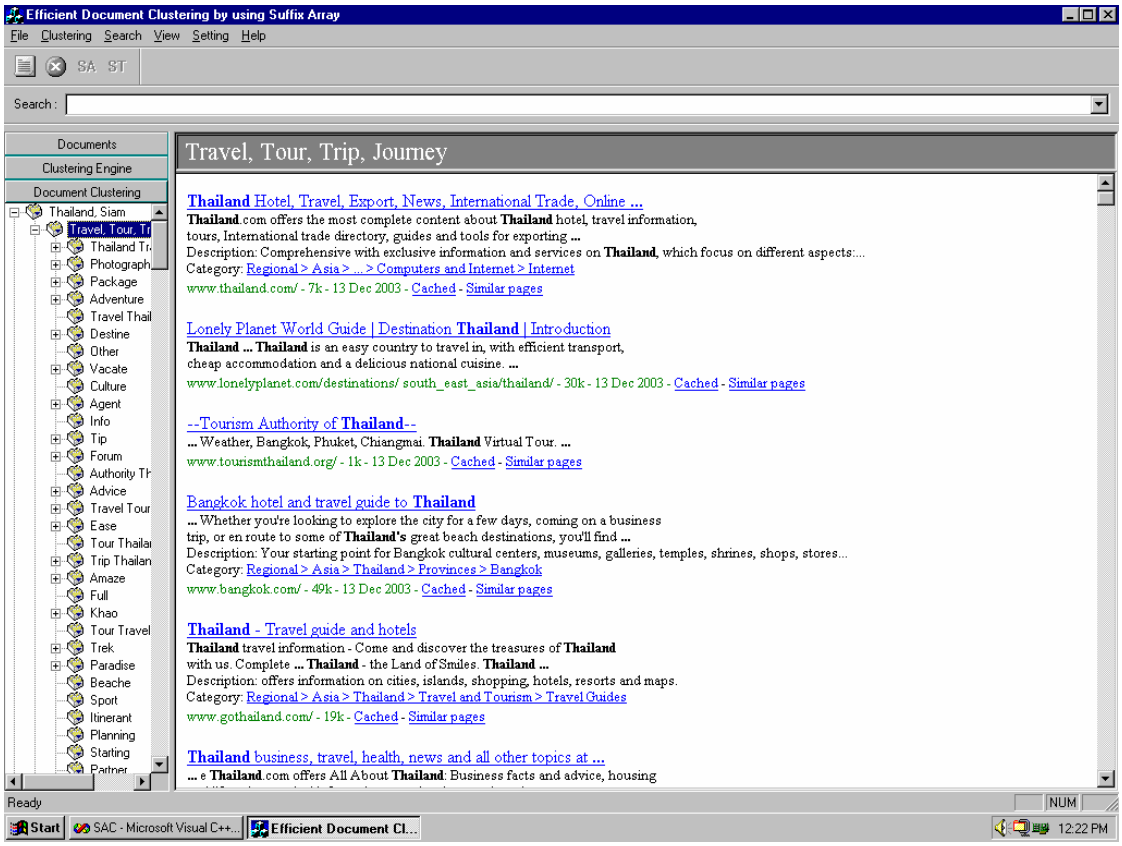


Figure 4.26: The results of document clustering in hierarchical format.

4.4 Summary

This chapter presented the detailed design and the implementation model of SAC prototype development including the system overview, the system design, details of all algorithms with flow diagrams or pseudocodes. In the next chapter, a complete prototype of SAC will be developed and tested in actual searching on the Web. The experimental results in various situations will be presented. The performance of SAC will be also evaluated.

CHAPTER V
EXPERIMENTAL RESULTS

5.1 Data Collection

In this chapter, the prototype of SAC will be tested on a real situation by using Web documents currently available. Its parameters will be studied. The experimental results will be presented and compared. The SAC performance will be also evaluated.

In our experiments, we use the Google search engine to retrieve the Web documents and use them as our test data for document clustering. The Google returns 20 titles of documents per page as a default value, but in our experiments, we select the numbers of search results (document titles) at 100, 200, 300, 400, 500, 600, 700, 800, 900 and 1000 titles to do experiments on SAC prototype. Figure 5.1 shows partial list of document titles returned from Google and used in our experiments.

The Stock Exchange of **Thailand**: Your Investment Resource for **...**

0-2229-2222 Email : SETCallCenter@set.or.th | Terms & Conditions of Use
| Privacy Policy Copyright © 2000 The Stock Exchange of **Thailand**. **...**
www.set.or.th/th/ - 20k - เก็บไว้แล้ว - หน้าคล้ายกัน

CIA - The World Factbook -- **Thailand**

... Introduction, **Thailand**, Top of Page. **...** Known as Siam until 1939, **Thailand** is the only Southeast Asian country never to have been taken over by a European power. **...**
www.cia.gov/cia/publications/factbook/geos/th.html - 101k - เก็บไว้แล้ว - หน้าคล้ายกัน

Figure 5.1: Partial list of search results returned by the search engine Google for a query “Thailand”.

Figure 5.1: Partial list of search results returned by the search engine Google for a query “Thailand” (cont.).

In addition to the data obtained from the Google search engine, we also get news information from the well-known news company website e.g., Reuters, CNN and BBC. These documents will be used for subsequent experiments on our SAC prototype.

5.2 Hardware and Software Environment

In the experiments, we use the hardware/software with the specifications shown in Table 5.1.

Table 5.1: Hardware/Software specifications.

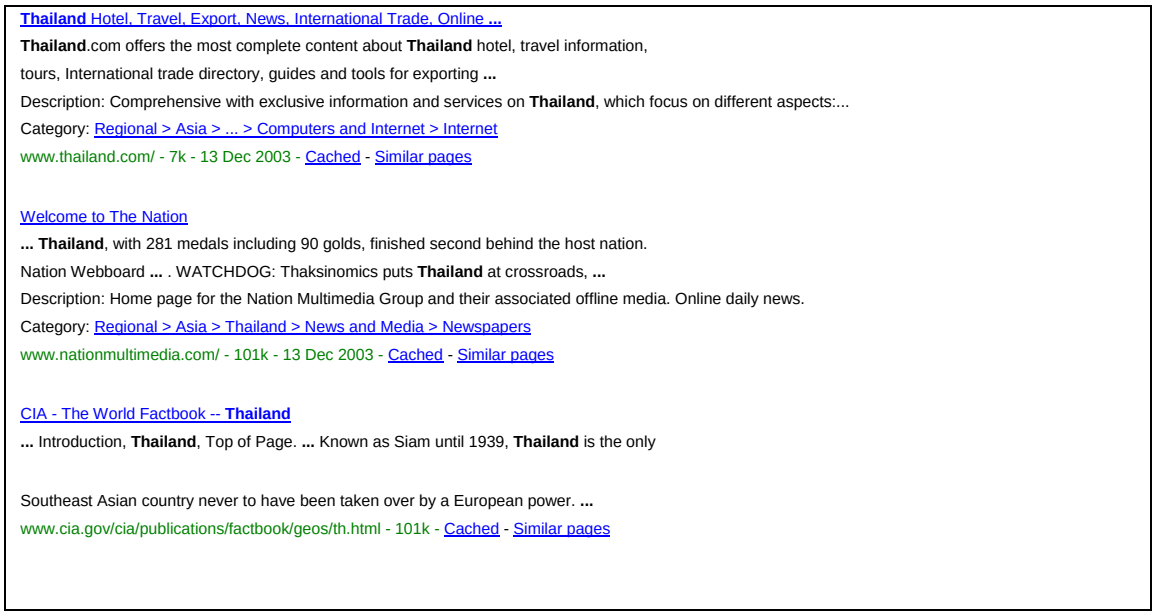
CPU	Pentium IV 400 MHz
RAM	256 MB
Hard Disk	40 GB
Operating System	Windows XP
Software Tools	MS Visual Studio 6.0
Programming Language	Visual C++ 6.0

5.3 Experiments and Results

In our SAC clustering technique, there are two principle parameters to be studied; T1, the threshold value for checking similarity and T2, the threshold value for assigning child or parent for phrase *i* and phrase *j* to display in hierarchy format as described in Chapter 3 and 4. Thus, many experiments will be conducted varying these two combinations of parameters. Normally, normalized similarity values are always positive and never go beyond one. However, the practical value that is reasonable to use for similarity value should not be less than 0.5 and we never use such a value as high as 1.0. In our experiments we use both threshold values of T1 and T2 ranging from 0.5 to 0.8.

We also test SAC on various sizes of document collections ranging from a 30-document collection up to a 1000-document collection.

We start the SAC experiment with a small collection containing 30 documents to show, in detail, the experimental results. A list of 30 documents is obtained from the Google search engine using a query keyword “Thailand” and is shown in Figure 5.2.



Thailand Hotel, Travel, Export, News, International Trade, Online **...**
Thailand.com offers the most complete content about **Thailand** hotel, travel information,
tours, International trade directory, guides and tools for exporting **...**
Description: Comprehensive with exclusive information and services on **Thailand**, which focus on different aspects:...
Category: Regional > Asia > ... > Computers and Internet > Internet
www.thailand.com/ - 7k - 13 Dec 2003 - Cached - Similar pages

Welcome to The Nation
... Thailand, with 281 medals including 90 golds, finished second behind the host nation.
Nation Webboard **...** . WATCHDOG: Thaksinomics puts **Thailand** at crossroads, **...**
Description: Home page for the Nation Multimedia Group and their associated offline media. Online daily news.
Category: Regional > Asia > Thailand > News and Media > Newspapers
www.nationmultimedia.com/ - 101k - 13 Dec 2003 - Cached - Similar pages

CIA - The World Factbook -- **Thailand**
... Introduction, **Thailand**, Top of Page. **...** Known as Siam until 1939, **Thailand** is the only

Southeast Asian country never to have been taken over by a European power. **...**
www.cia.gov/cia/publications/factbook/geos/th.html - 101k - Cached - Similar pages

Figure 5.2: A list of 30 documents obtained from the Google search engine using a query keyword “Thailand”.

Thailand - WWW Virtual Library: **Thailand** the Big Picture
Go to **Thailand** The Big Picture Home, **Thailand** WWW Virtual Library. **...** It has been created to keep track of leading information facilities in/about **Thailand**. **...**
www.nectec.or.th/WWW-VL-Thailand.html - 16k - 13 Dec 2003 - Cached - Similar pages

Bank of **Thailand**
... Best View with MSIE4.0+ 800x600 Resolution with medium font (Technical Recommendations).
Copyright © 1995 Bank of **Thailand**. All rights reserved. **...**

Description: Decisions of the Monetary Policy Committee, guide to financial investment in **Thailand**, debt issuance...
Category: Society > Government > Finance > Central Banks
www.bot.or.th/ - 52k - 13 Dec 2003 - Cached - Similar pages

Lonely Planet World Guide | Destination **Thailand** | Introduction
Thailand ... Thailand is an easy country to travel in, with efficient transport, cheap accommodation and a delicious national cuisine. **...**
www.lonelyplanet.com/destinations/ south_east_asia/thailand/ - 30k - 13 Dec 2003 - Cached - Similar pages

Lonely Planet - **Thailand** Map
home.
www.lonelyplanet.com/mapshells/south_east_asia/ thailand/thailand.htm - 6k - Cached - Similar pages
[More results from www.lonelyplanet.com]

--Tourism Authority of **Thailand**--
... Weather, Bangkok, Phuket, Chiangmai. **Thailand** Virtual Tour. **...**
www.tourismthailand.org/ - 1k - 13 Dec 2003 - Cached - Similar pages

Ministry of Foreign Affairs, Kingdom of **Thailand**
Description: Information related to foreign policy, consular and economic affairs, protocol and news updates.
Category: Regional > Asia > Thailand > Government
www.mfa.go.th/ - 1k - 13 Dec 2003 - Cached - Similar pages

Bangkok Post Main Entrance Page
... Investigations set to pick up The Stock Exchange of **Thailand** plans to forward evidence of share manipulation occurring over the second and third quarters to **...**
Description: Bangkok's daily newspaper in English language.
Category: Regional > Asia > Thailand > Provinces > Bangkok
www.bangkokpost.net/ - 40k - Cached - Similar pages

Thailand Information at Mahidol University, **Thailand**
Welcome to **Thailand**. The Land of Smiles. Sawasdee Ka! Our site has been awarded 3 stars by Magellan three stars. Search **Thailand** Contents. **...**
www.mahidol.ac.th/Thailand/Thailand-main.html - 14k - 13 Dec 2003 - Cached - Similar pages

Bangkok hotel and travel guide to **Thailand**
... Whether you're looking to explore the city for a few days, coming on a business trip, or en route to some of **Thailand's** great beach destinations, you'll find **...**
Description: Your starting point for Bangkok cultural centers, museums, galleries, temples, shrines, shops, stores...
Category: Regional > Asia > Thailand > Provinces > Bangkok
www.bangkok.com/ - 49k - 13 Dec 2003 - Cached - Similar pages

Figure 5.2: A list of 30 documents obtained from the Google search engine using a query keyword “Thailand” (cont.).

Internet **Thailand** - The first ISP in **Thailand**
Established in March 1995, Internet **Thailand** is the country's leading
ISP for both individual and corporate users. International **...**
Description: Offers both individual and corporate user accounts. International bandwidth totals 10Mbps with fiber...
Category: Regional > Asia > ... > Internet > Access Providers
www.inet.co.th/ - 74k - 13 Dec 2003 - Cached - Similar pages

THNIC : **Thailand** Network Information Center
Description: NIC for .th CCTLD.
Category: Computers > Internet > ... > Country Domains > T
www.thnic.net/ - 1k - Cached - Similar pages

Bangkok Post Main Entrance Page
... Plc successfully raised 22.15 billion baht by selling out its public offering despite
poor market conditions in which the Stock Exchange of **Thailand** index fell **...**
www.bangkokpost.com/ - 39k - Cached - Similar pages

Investment in **Thailand**: Business and Investment Information for **...**
The Thai Board of Investment (BOI) - your investment and business information source
for **Thailand**. **... Thailand** investment information for international investors. **...**
Description: The Thai Board of Investment (BOI) - your investment and business information source for **Thailand**.
Category: Regional > Asia > ... > Government > Office of the Prime Minister
www.boi.go.th/ - 4k - 13 Dec 2003 - Cached - Similar pages

The Royal Thai Embassy, Washington, DC
Description: Washington, DC.
Category: Regional > Asia > ... > Government > Embassies and Consulates
www.thaiembdc.org/ - 1k - Cached - Similar pages

Thailand Hotels: Hotels in **Thailand**, **Thailand** Hotels Reservation **...**
Hotels reservations in all **Thailand** cities: Bangkok, Cha-Am, Chiang Mai, Chiang
Rai, Hua Hin, Krabi, Pattaya, Phuket, Samui. **...** Best Prices on **Thailand** Hotels. **...**
www.thailand-hotelsonline.com/ - 12k - Cached - Similar pages

US Embassy in **Thailand**
The United States Mission in **Thailand**: US Embassy in Bangkok, Consulate
in Chiang Mai, United States Information Service. CURRENT ISSUES: **...**
www.usa.or.th/ - 13k - 13 Dec 2003 - Cached - Similar pages

National Statistical Office **Thailand**
... Latest Release Reports. Key Statistics of **Thailand** 2001. Ongoing Census.
2002 Business Trade and Services Census. .. Indicators of **Thailand**. **...**
www.nso.go.th/eng/ - 14k - Cached - Similar pages

National Statistical Office **Thailand**
... Latest Release Reports. Key Statistics of **Thailand** 2003. Ongoing Census.
2002 Business Trade and Services Census. .. Indicators of **Thailand**. **...**
Description: Statistical site providing economic, agriculture, industry and socio-economic data.
Category: Science > Social Sciences > ... > Official Statistics > Asia
www.nso.go.th/ - 14k - Cached - Similar pages
[More results from www.nso.go.th]

CAT Telecom Public Company Limited
The link that you are following will take you to a non-cattelecom.com
location. To continue click here.
Description: **Thailand**
Category: Society > Government > ... > Postal Services
www.cat.or.th/ - 1k - Cached - Similar pages

Figure 5.2: A list of 30 documents obtained from the Google search engine using a query keyword “Thailand” (cont.).

Thailand - Travel guide and hotels
Thailand travel information - Come and discover the treasures of **Thailand**
with us. Complete **... Thailand** - the Land of Smiles. **Thailand ...**
Description: offers information on cities, islands, shopping, hotels, resorts and maps.
Category: Regional > Asia > Thailand > Travel and Tourism > Travel Guides
www.gothailand.com/ - 19k - Cached - Similar pages

Library of Congress / Federal Research Division / Country Studies **...**
THAILAND - A Country Study. Search **Thailand** Include word variants Use
only words as entered. **THAILAND**; Foreward; Acknowledgments; Preface; **...**
memory.loc.gov/frd/cs/thtoc.html - 19k - Cached - Similar pages

Thailand Environment Institute
... We are moving... From 17 November 2003, the **Thailand** Environment Institute will
move to the new office. **...** Copyright 2002 **Thailand** Environment Institute. **...**
Description: Non-profit, non-governmental organization focused on environmental management.
Category: Regional > Asia > ... > Science and Environment > Organizations
www.tei.or.th/ - 8k - Cached - Similar pages

Weather Underground: Bangkok, **Thailand** Forecast
... Click Here. Bangkok, **Thailand** this page to your Favorites, 13.9° N 100.6° E | MapBlast. **...** Updated:
10:30 AM ICT on December 14, 2003 Observed at Bangkok, **Thailand**. **...**
www.wunderground.com/global/stations/48456.html - 41k - 13 Dec 2003 - Cached - Similar pages

Weather Underground: **Thailand**
... Click Here. Search results for: **Thailand**. = Weather Warning. to Favorites.
Place, Temperature, Humidity, Pressure, Conditions, Wind, Updated, **...**
www.wunderground.com/global/TH.html - 47k - 13 Dec 2003 - Cached - Similar pages
[More results from www.wunderground.com]

Thailand business, travel, health, news and all other topics at **...**
... e **Thailand**.com offers All About **Thailand**: Business facts and advice, housing
and lifestyle, practical information, travel, culture and much more. **...**
Description: Community portal offering email addresses, a newsletter, classified advertisements, a links directory...
Category: Regional > Asia > Thailand > Guides and Directories
www.ethailand.com/ - 74k - 13 Dec 2003 - Cached - Similar pages

Web design **Thailand** - Webdesign and Logo
Web design, webdesign, logo, website design in Phuket, **Thailand** - Advertising company,
offering professional services in graphics and corporate logo creation. **...**
Description: Advertising company, offering professional services in Website and graphic design, desktop publishing...
Category: Computers > Internet > ... > Designers > Full Service > A
www.andagraf.com/ - 13k - Cached - Similar pages

thailand hotels - bangkok hotel - phuket hotel - pattaya hotels
Thailand Hotels - Bangkok Hotel - Phuket Hotel Reservation. **...** Daily update discount
tariff. **thailand** hotels - bangkok hotel - phuket hotel - pattaya hotels. **...**
www.siam.net/ - 83k - Cached - Similar pages

Figure 5.2: A list of 30 documents obtained from the Google search engine using a query keyword “Thailand” (cont.).

When a complete list of documents is received, SAC starts its process with lexical analysis. Table 5.2 shows a list of words/tokens produced by SAC lexical analyzer.

Table 5.2: A list of words/tokens parsed from documents.

about	come	explore	ict	manipulate	place	siam	underground
accommodate	coming	export	include	map	plan	site	unite
acknowledge	company	exporting	including	mapblast	planet	smile	university
advertising	complete	facile	index	march	plc	some	until
advice	condition	fact	indicate	market	poor	source	update
affair	congress	factbook	individual	medal	post	southeast	use
all	consul	fall	inform	medium	power	star	user
and	contain	favor	institute	minister	practice	state	vary
are	continue	federate	international	mission	preface	statistics	view
asia	copyright	few	internet	more	press	stock	virtual
authority	corporation	find	introduce	most	price	study	warn
award	country	finish	invest	move	profess	succeed	washington
baht	create	first	investigate	much	publish	take	watchdog
bangkok	crossroad	follow	isp	nation	puts	taken	weather
bank	cuisine	font	issue	network	quarter	tariff	web
beach	culture	for	its	never	rai	technics	webboard
been	current	forecast	keep	new	raise	telecom	webdesign
behind	daily	foreign	key	news	recommend	temper	website
best	days	foreward	king	non	release	thailand	welcome
big	december	forward	know	november	report	thaksinomic	whether
billion	delicate	from	krabi	observe	research	that	which
board	design	gold	land	occurring	reservation	the	will
boi	despite	graphic	late	offer	reserve	third	wind
both	destine	great	lead	office	resolve	this	with
busy	direct	guide	library	ongoing	result	thnic	word
cat	discount	has	lifestyle	online	right	three	world
census	discover	have	limit	only	route	tool	www
center	divide	heal	link	other	royal	top	you
cha	ease	here	local	our	samui	topic	your
change	efficient	hin	logo	out	sawasdee	tour	
cheap	embassy	home	lone	over	search	track	
chiang	enter	host	looking	page	second	trade	
chiangmai	environ	hotel	magellan	pattaya	selling	transport	
cia	establish	house	mahidol	phuket	serve	travel	
city	european	hua	mai	pick	set	treasure	
click	evidence	humid	main	picture	share	trip	

The list of words/tokens output from lexical analyzer is, then, passed to the process of stopword elimination. Table 5.3 shows stopwords found in document collection and were removed. Then, SAC proceeds to the stemming process. Table 5.4 shows a list of words remaining after stemming.

Table 5.3: Removed stopwords

About	best	Few	its	november	some	until	you
all	both	For	march	only	take	use	your
and	come	From	more	our	taken	whether	
are	coming	Has	most	out	that	which	
been	days	Have	much	over	the	will	
behind	december	Here	never	puts	this	with	

Table 5.4: Remaining words after stemming process.

Word	Stem	Word	Stem
accommodation	accommodate	leading	lead
acknowledgments	acknowledge	limited	limit
affairs	affair	location	local
asian	asia	lonely	lone
awarded	award	manipulation	manipulate
business	busy	medals	medal
cities	city	ministry	minister
conditions	condition	moving	move
consulate	consul	national	nation
content	contain	observed	observe
contents	contain	offering	offer
corporate	corporation	offers	offer
created	create	plans	plan
creation	create	practical	practice
crossroads	crossroad	pressure	press
delicious	delicate	prices	price
destination	destine	professional	profess
destinations	destine	public	publish
directory	direct	quarters	quarter
division	divide	raised	raise
easy	ease	recommendations	recommend
entered	enter	reports	report
entrance	enter	reservations	reservation
environment	environ	reserved	reserve
established	establish	resolution	resolve

exchange	change	results	result
facilities	facile	rights	right
facts	fact	service	serve
favorites	favor	services	serve
federal	federate	smiles	smile
fell	fall	stars	star
finished	finish	states	state
following	follow	statistical	statistics
golds	gold	studies	study
graphics	graphic	successfully	succeed
guides	guide	technical	technics
health	heal	temperature	temper
hotels	hotel	thai	thailand
housing	house	thaksinomics	thaksinomic
humidity	humid	tools	tool
indicators	indicate	topics	topic
information	inform	tourism	tour
introduction	introduce	tours	tour
investigations	investigate	treasures	treasure
investment	invest	united	unite
investors	invest	updated	update
issues	issue	users	user
kingdom	king	variants	vary
known	know	warning	warn
latest	late	words	word

When stemming is completed, SAC starts the process of clustering by counting each word frequency for each document. Table 5.5 shows a frequency table of words in documents.

Table 5.5: Frequencies of words in documents.

Words	Document no.																													
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
thailand	2	2	3	5	2	3	1	2	1	1	4	1	3	1	1	4	1	5	2	3	3		5	3	3	3	2	2	2	3
bangkok								1		1		1			1			1	1							3				3
inform	1			1							1			1		3			1				1					1		
hotel	2											1						5					1							11
travel	2					1						1											2					2		
busy												1				2				1	1							2		
nation		3				1														1	1									
serve																			1	1	1								1	
phuket								1										1										1	1	
offer	1														1															
page			1							1					1											1				
guide	1					1						1											1							
international	2												1			1														
search											1													1			1			
click																						1				1	1			
update																										1	1			1
enter										1					1									1						
weater								1																		1	2			
office																				1	1				1					
trade	2																			1	1									
thailand hotel	2																	3												3
country			1			1																		2						

With the threshold value of T1=0.8 and T2=0.8, we get the results of clustering shown in Figure 5.3.

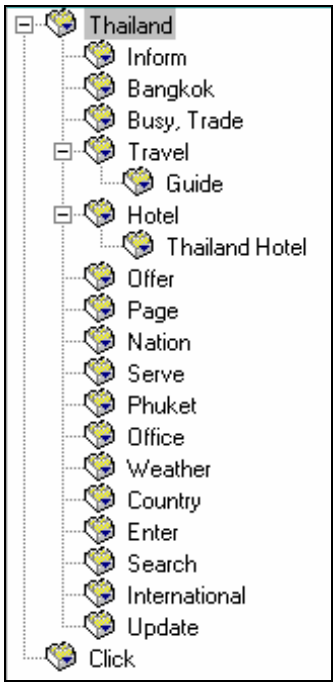


Figure 5.3: Hierarchical structure of documents clustered for T1=0.8, T2=0.8, collection size=30.

We check the correctness of clustering results by simple verifying manually the correctness of the right documents to be put in the right clusters. In this experiment, we found that 23 documents are put in the right clusters and 7 documents are put in the wrong clusters. This implies the clustering precision of 23/30 = 0.76.

Next, we conduct more experiments using various combination of threshold values T1 and T2 as well as the sizes of document collections. Table 5.6 - 5.30 and Figure 5.4 – 5.28 show the summary results of all experiments conducted.

Table 5.6: Combining base clusters for T1=0.8, T2=0.8 and collection size=100.

Word1	No. of doc. found	Word2	No. of doc. found	No. of doc. in Union	No. of doc. in Intersect	Sim.
mai	8	chiang mai	7	8	7	0.875
accommodate guide	3	Reservation accommodate	3	3	3	1.000

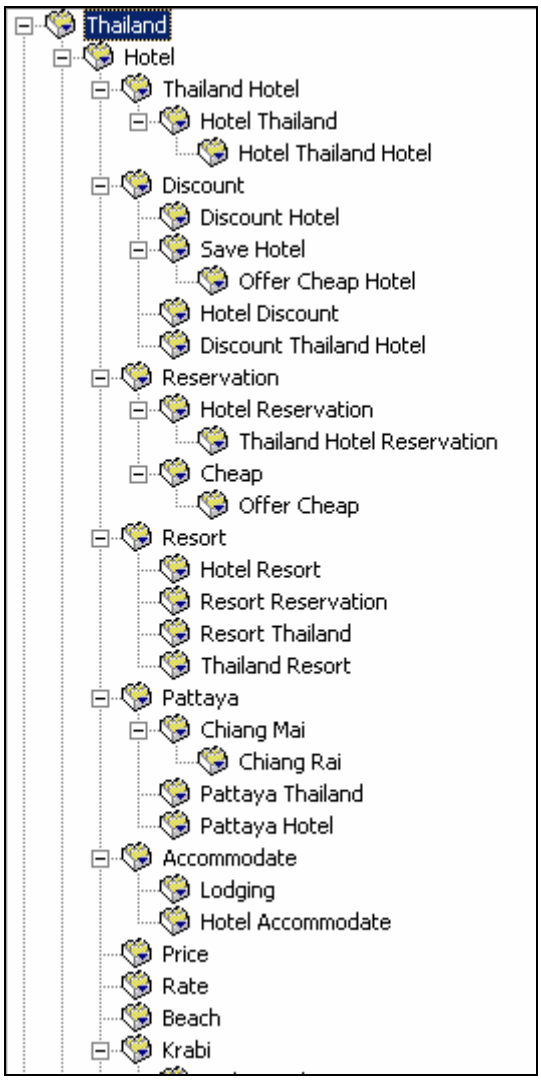


Figure 5.4: Clustering results for T1=0.8, T2=0.8 and collection size=100.

Table 5.7: Combining base clusters for T1=0.8, T2=0.7 and collection size=100.

Word1	No. of doc. found	Word2	No. of doc. found	No. of doc. in Union	No. of doc. in Intersect	Sim.
mai	8	chiang mai	7	8	7	0.875
accommodate guide	3	Reservation accommodate	3	3	3	1.000

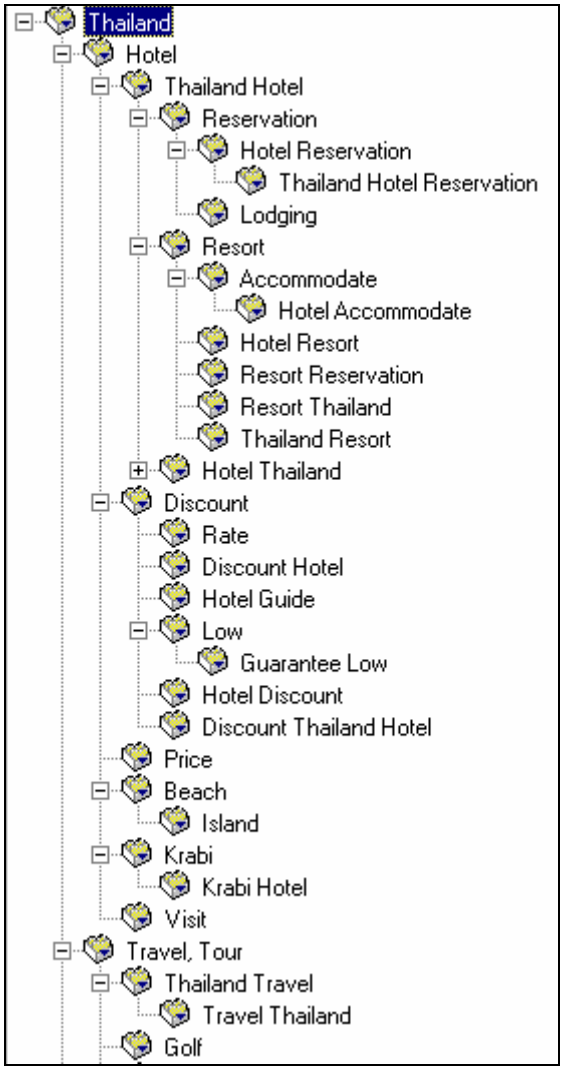


Figure 5.5: Clustering results for T1=0.8, T2=0.7 and collection size=100.

Table 5.8: Combining base clusters for T1=0.8, T2=0.6 and collection size=100.

Word1	No. of doc. found	Word2	No. of doc. found	No. of doc. in Union	No. of doc. in Intersect	Sim.
mai	8	chiang mai	7	8	7	0.875
accommodate guide	3	Reservation accommodate	3	3	3	1.000

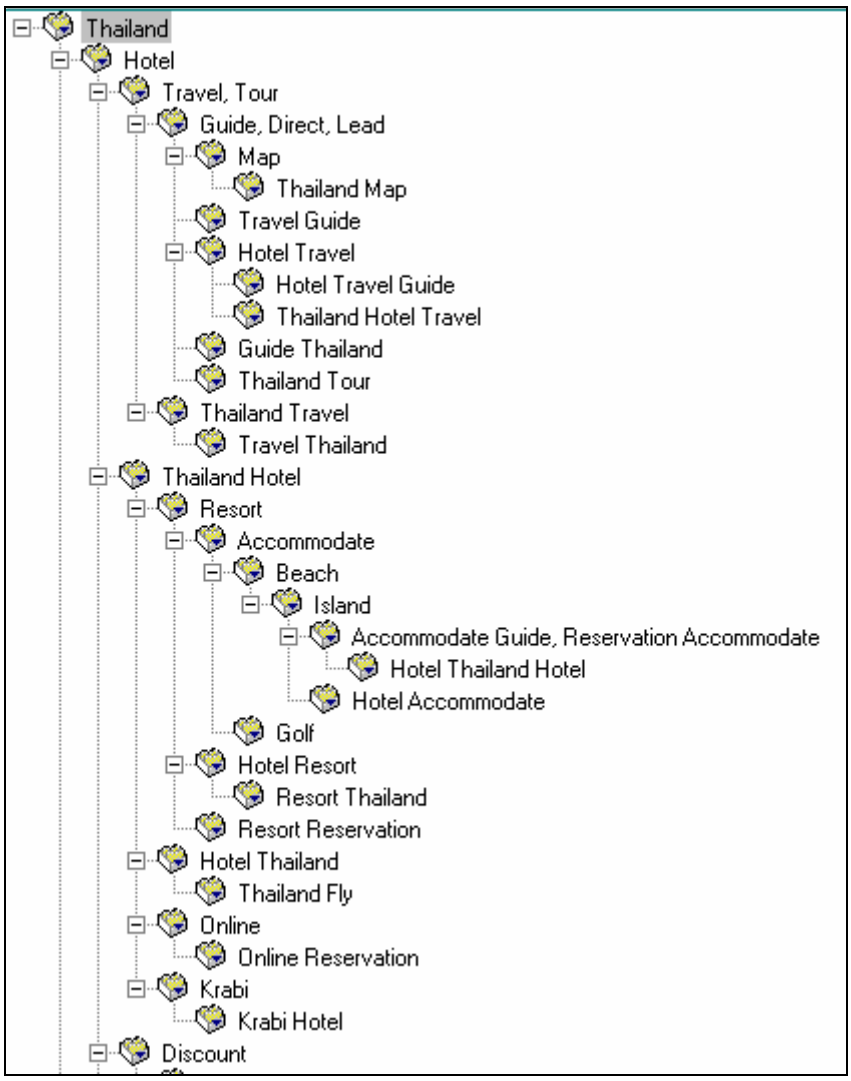


Figure 5.6: Clustering results for T1=0.8, T2=0.6 and collection size=100.

Table 5.9: Combining base clusters for T1=0.8, T2=0.5 and collection size=100.

Word1	No. of doc. found	Word2	No. of doc. found	No. of doc. in Union	No. of doc. in Intersect	Sim.
mai	8	chiang mai	7	8	7	0.875
accommodate guide	3	Reservation accommodate	3	3	3	1.000

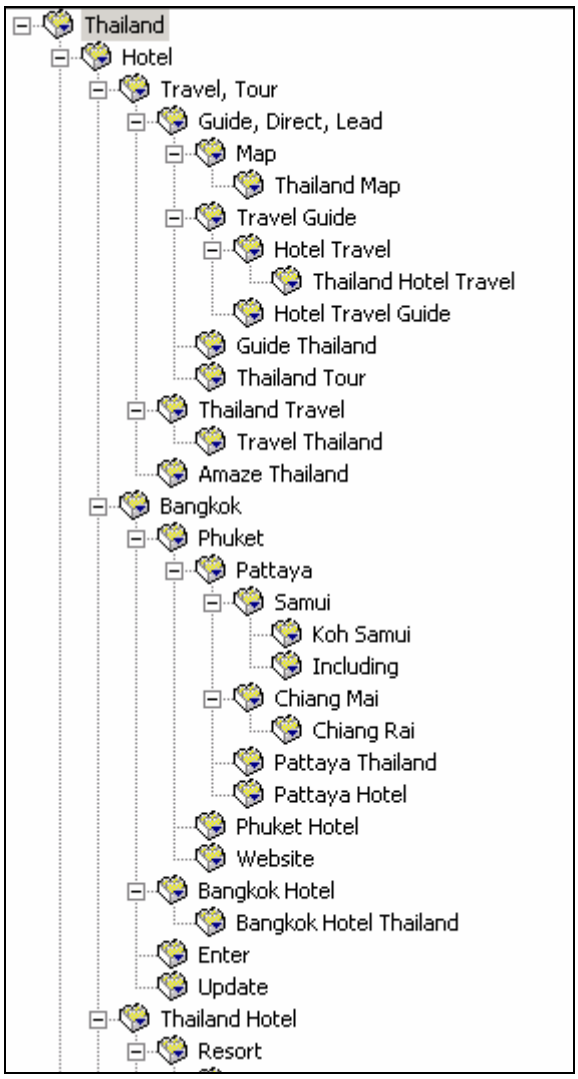


Figure 5.7: Clustering results for T1=0.8, T2=0.5 and collection size=100.

Table 5.10: Combining base clusters for T1=0.7, T2=0.8 and collection size=100.

Word1	No. of doc. found	Word2	No. of doc. found	No. of doc. in Union	No. of doc. in Intersect	Sim.
mai	8	chiang mai	7	8	7	0.875
cheap	7	offer cheap	5	7	5	0.714
save hotel	4	offer cheap hotel	3	4	3	0.750
low	4	guarantee low	3	4	3	0.750
change	4	stock change thailand	3	4	3	0.750
accommodate guide	3	Reservation accommodate	3	3	3	1.000

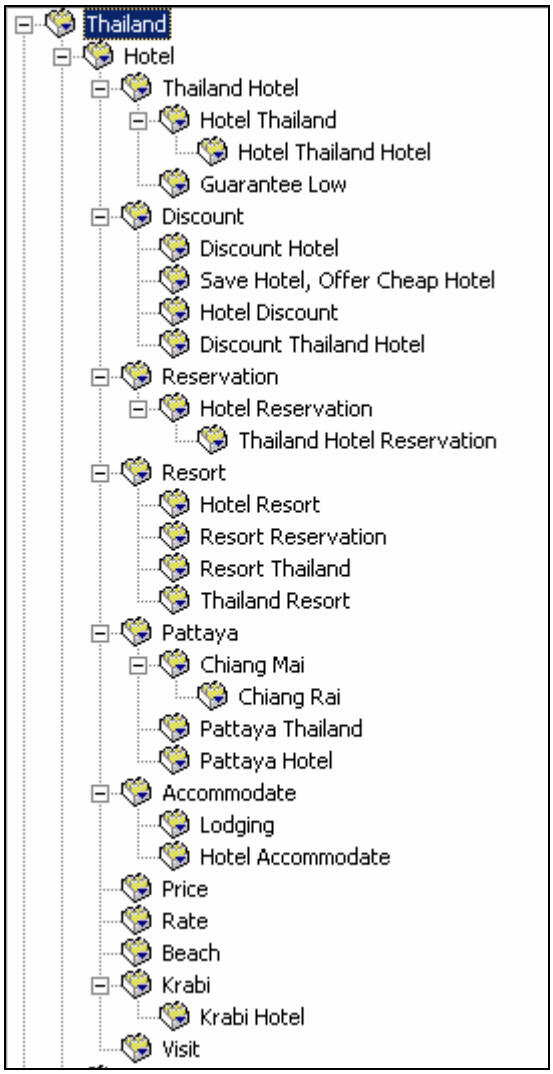


Figure 5.8: Clustering results for T1=0.7, T2=0.8 and collection size=100.

Table 5.11: Combining base clusters for T1=0.7, T2=0.7 and collection size=100.

Word1	No. of doc. found	Word2	No. of doc. found	No. of doc. in Union	No. of doc. in Intersect	Sim.
mai	8	chiang mai	7	8	7	0.875
cheap	7	offer cheap	5	7	5	0.714
save hotel	4	offer cheap hotel	3	4	3	0.750
low	4	guarantee low	3	4	3	0.750
change	4	stock change thailand	3	4	3	0.750
accommodate guide	3	Reservation accommodate	3	3	3	1.000

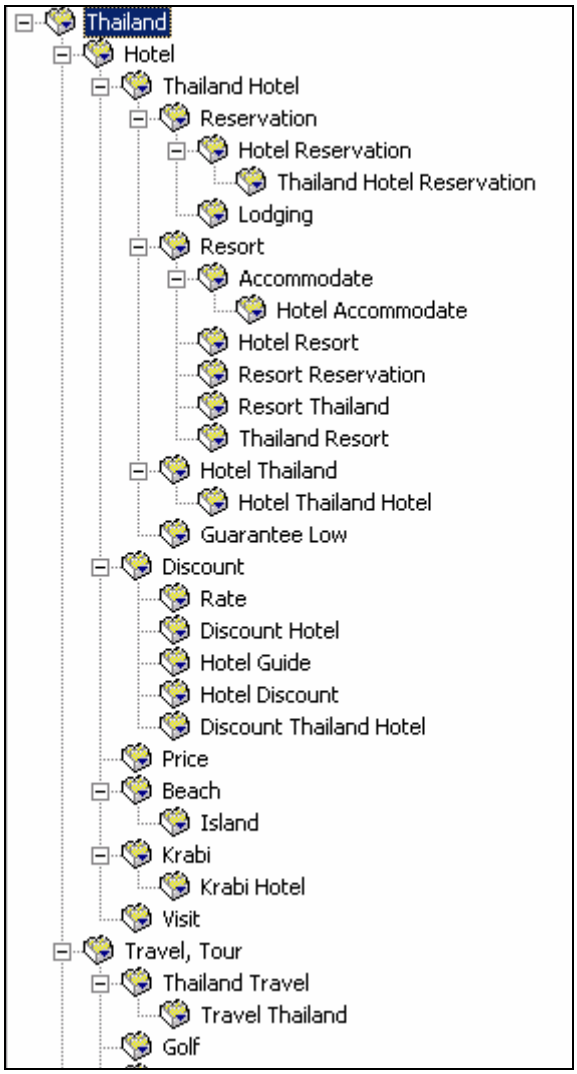


Figure 5.9: Clustering results for T1=0.7, T2=0.7 and collection size=100.

Table 5.12: Combining base clusters for T1=0.7, T2=0.6 and collection size=100.

Word1	No. of doc. found	Word2	No. of doc. found	No. of doc. in Union	No. of doc. in Intersect	Sim.
mai	8	chiang mai	7	8	7	0.875
cheap	7	offer cheap	5	7	5	0.714
save hotel	4	offer cheap hotel	3	4	3	0.750
low	4	guarantee low	3	4	3	0.750
change	4	stock change thailand	3	4	3	0.750
accommodate guide	3	Reservation accommodate	3	3	3	1.000

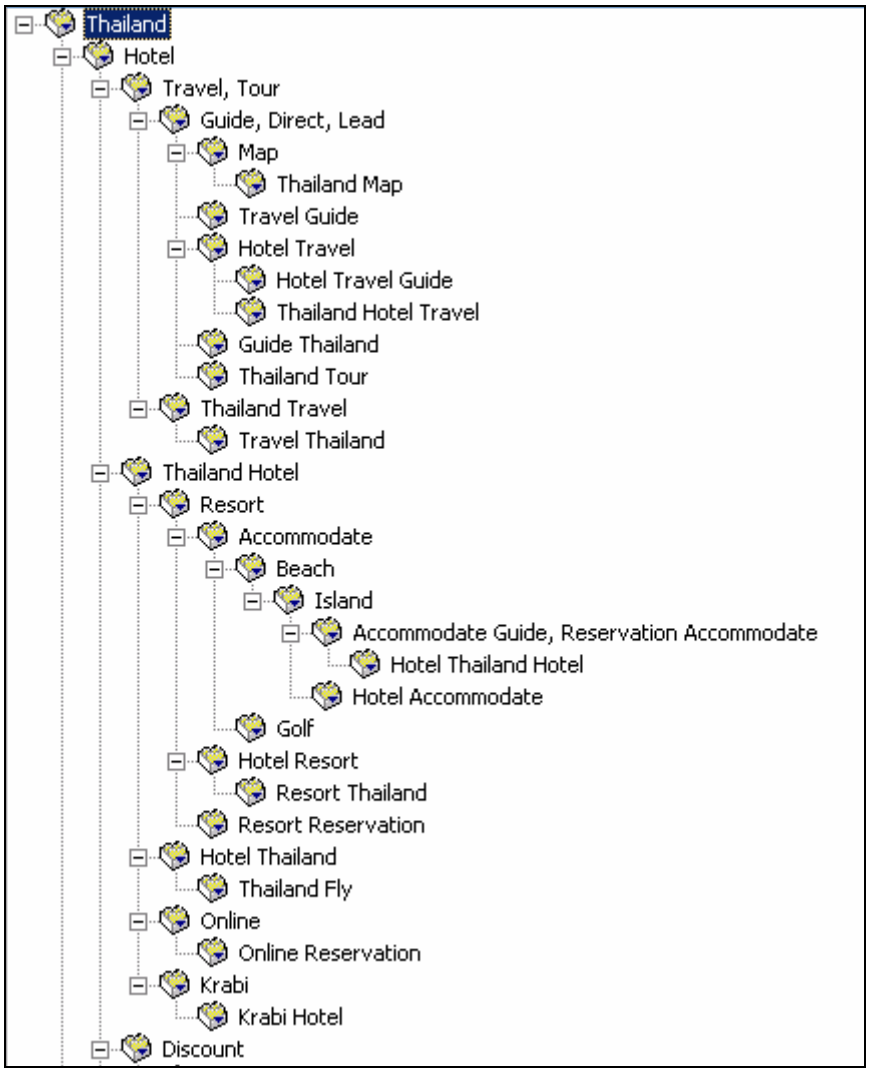


Figure 5.10: Clustering results for T1=0.7, T2=0.6 and collection size=100.

Table 5.13: Combining base clusters for T1=0.7, T2=0.5 and collection size=100.

Word1	No. of doc. found	Word2	No. of doc. found	No. of doc. in Union	No. of doc. in Intersect	Sim.
mai	8	chiang mai	7	8	7	0.875
cheap	7	offer cheap	5	7	5	0.714
save hotel	4	offer cheap hotel	3	4	3	0.750
low	4	guarantee low	3	4	3	0.750
change	4	stock change thailand	3	4	3	0.750
accommodate guide	3	Reservation accommodate	3	3	3	1.000

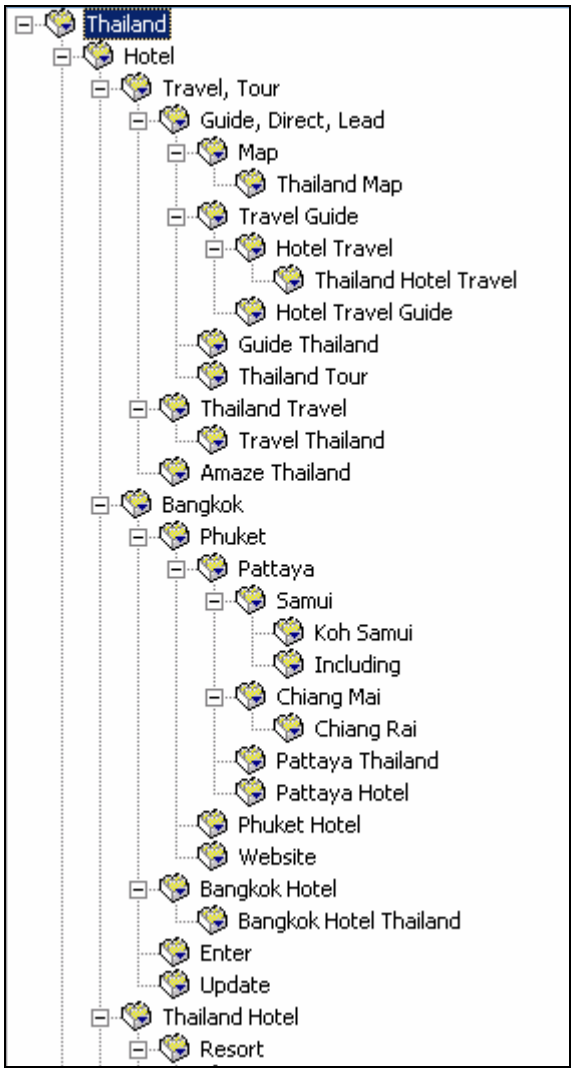


Figure 5.11: Clustering results for T1=0.7, T2=0.5 and collection size=100.

Table 5.14: Combining base clusters for T1=0.6, T2=0.8 and collection size=100.

Word1	No. of doc. found	Word2	No. of doc. found	No. of doc. in Union	No. of doc. in Intersect	Sim.
hotel	41	thailand hotel	26	41	26	0.634
mai	8	chiang mai	7	8	7	0.875
cheap	7	offer cheap	5	7	5	0.714
save hotel	4	offer cheap hotel	3	4	3	0.750
low	4	guarantee low	3	4	3	0.750
change	4	stock change thailand	3	4	3	0.750
accommodate guide	3	Reservation accommodate	3	3	3	1.000

Figure 5.12: Clustering results for T1=0.6 T2=0.8and collection size=100.

Table 5.15: Combining base clusters for T1=0.6, T2=0.7 and collection size=100.

Word1	No. of doc. found	Word2	No. of doc. found	No. of doc. in Union	No. of doc. in Intersect	Sim.
hotel	41	thailand hotel	26	41	26	0.634
mai	8	chiang mai	7	8	7	0.875
cheap	7	offer cheap	5	7	5	0.714
save hotel	4	offer cheap hotel	3	4	3	0.750
low	4	guarantee low	3	4	3	0.750
change	4	stock change thailand	3	4	3	0.750
accommodate guide	3	Reservation accommodate	3	3	3	1.000

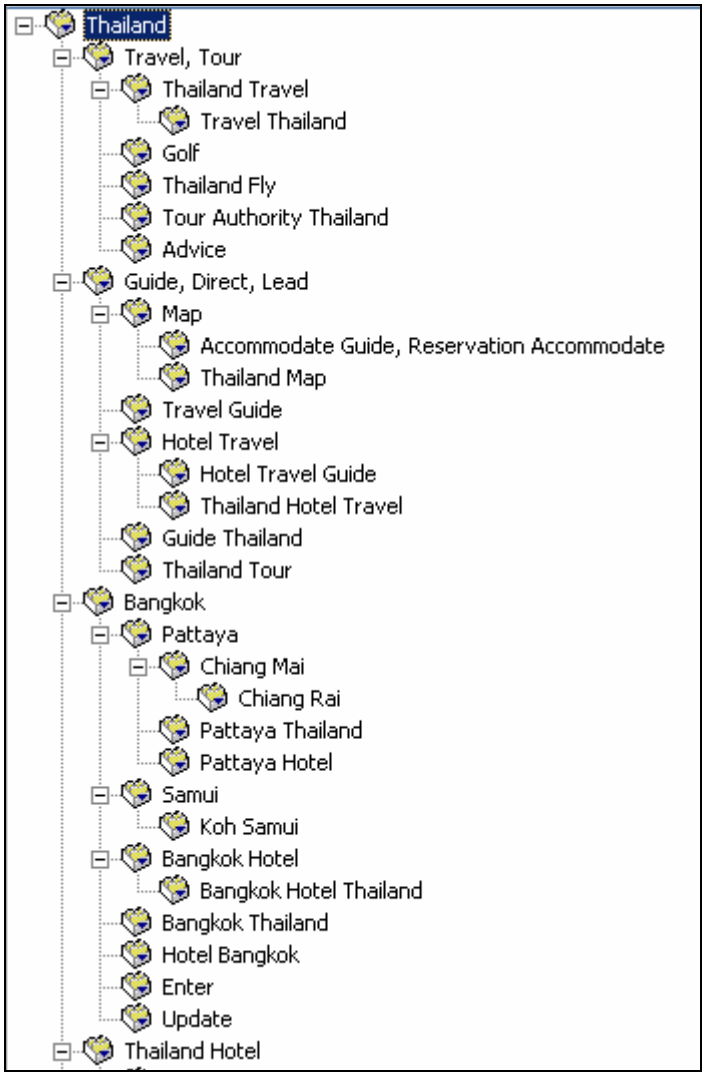


Figure 5.13: Clustering results for T1=0.6 T2=0.7 and collection size=100.

Table 5.16: Combining base clusters for T1=0.6, T2=0.6 and collection size=100.

Word1	No. of doc. found	Word2	No. of doc. found	No. of doc. in Union	No. of doc. in Intersect	Sim.
hotel	41	thailand hotel	26	41	26	0.634
mai	8	chiang mai	7	8	7	0.875
cheap	7	offer cheap	5	7	5	0.714
save hotel	4	offer cheap hotel	3	4	3	0.750
low	4	guarantee low	3	4	3	0.750
change	4	stock change thailand	3	4	3	0.750
accommodate guide	3	Reservation accommodate	3	3	3	1.000

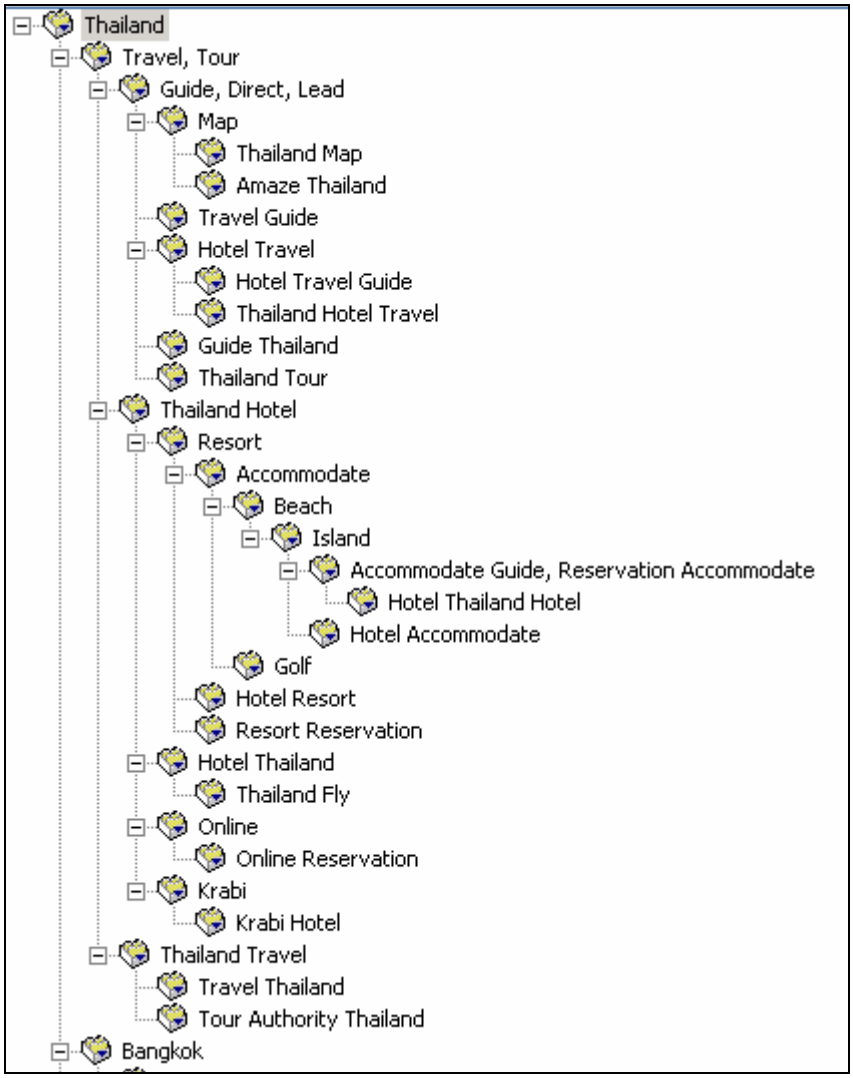


Figure 5.14: Clustering results for T1=0.6 T2=0.6 and collection size=100.

Table 5.17: Combining base clusters for T1=0.6, T2=0.5 and collection size=100.

Word1	No. of doc. found	Word2	No. of doc. found	No. of doc. in Union	No. of doc. in Intersect	Sim.
hotel	41	thailand hotel	26	41	26	0.634
mai	8	chiang mai	7	8	7	0.875
cheap	7	offer cheap	5	7	5	0.714
save hotel	4	offer cheap hotel	3	4	3	0.750
low	4	guarantee low	3	4	3	0.750
change	4	stock change thailand	3	4	3	0.750
accommodate guide	3	Reservation accommodate	3	3	3	1.000

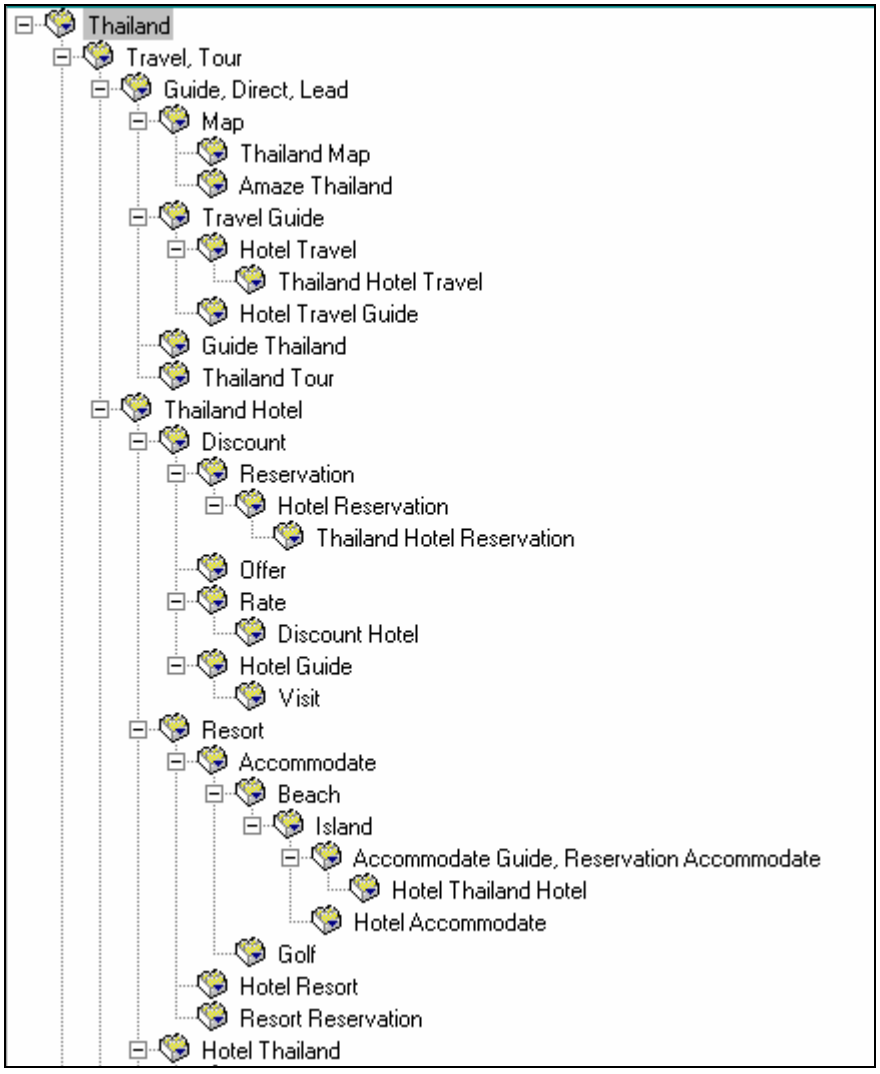


Figure 5.15: Clustering results for T1=0.6 T2=0.5 and collection size=100.

Table 5.18: Combining base clusters for T1=0.5, T2=0.8 and collection size=100.

Word1	No. of doc. found	Word2	No. of doc. found	No. of doc. in Union	No. of doc. in Intersect	Sim.
hotel	41	thailand hotel	26	41	26	0.634
travel	28	guide	22	33	17	0.515
hotel reservation	9	thailand hotel reservation	5	9	5	0.555
mai	8	chiang mai	7	8	7	0.875
cheap	7	offer cheap	5	7	5	0.714
hotel travel	5	thailand hotel travel	3	5	3	0.600
hotel travel	5	hotel travel guide	3	5	3	0.600
save hotel	4	offer cheap hotel	3	4	3	0.750
low	4	guarantee low	3	4	3	0.750
change	4	stock change thailand	3	4	3	0.750
accommodate guide	3	reservation accommodate	3	3	3	1.000

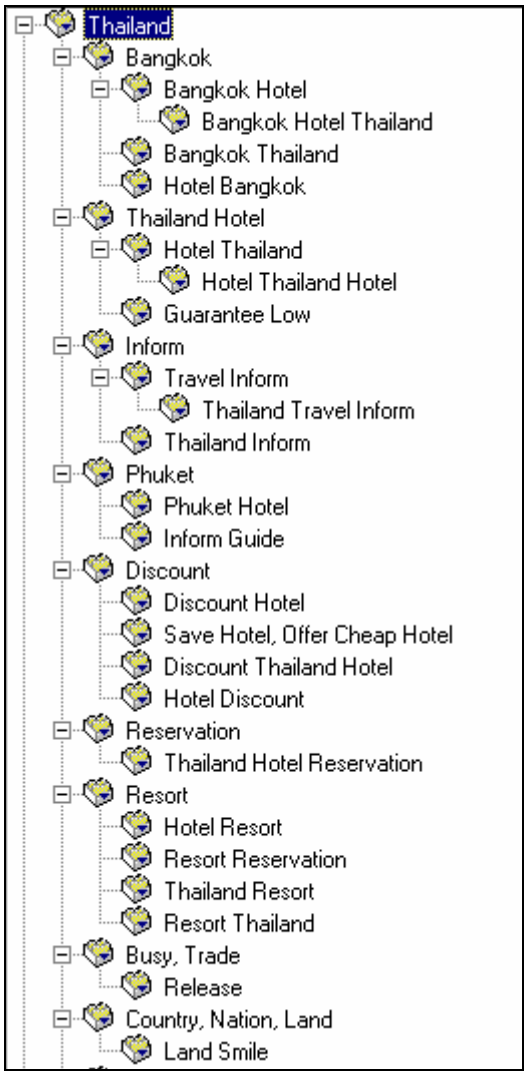


Figure 5.16: Clustering results for T1=0.5 T2=0.8 and collection size=100.

Table 5.19: Combining base clusters for T1=0.5, T2=0.7 and collection size=100.

Word1	No. of doc. found	Word2	No. of doc. found	No. of doc. in Union	No. of doc. in Intersect	Sim.
hotel	41	thailand hotel	26	41	26	0.634
travel	28	guide	22	33	17	0.515
hotel reservation	9	thailand hotel reservation	5	9	5	0.555
mai	8	chiang mai	7	8	7	0.875
cheap	7	offer cheap	5	7	5	0.714
hotel travel	5	thailand hotel travel	3	5	3	0.600
hotel travel	5	hotel travel guide	3	5	3	0.600
save hotel	4	offer cheap hotel	3	4	3	0.750
low	4	guarantee low	3	4	3	0.750
change	4	stock change thailand	3	4	3	0.750
accommodate guide	3	reservation accommodate	3	3	3	1.000

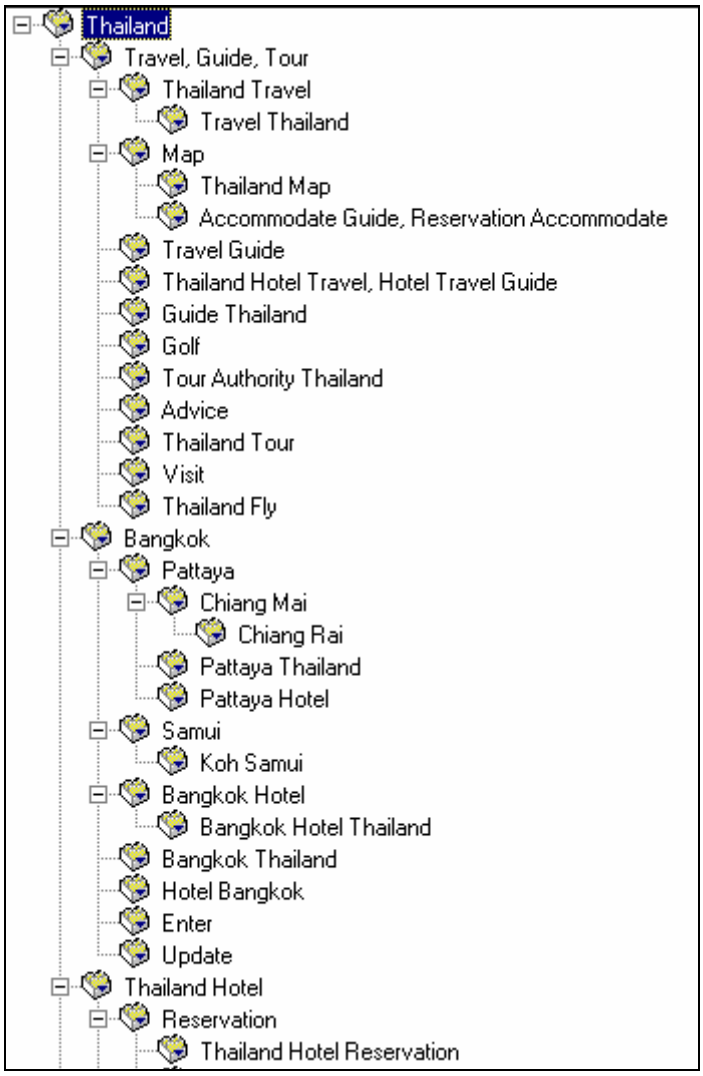


Figure 5.17: Clustering results for T1=0.5 T2=0.7 and collection size=100.

Table 5.20: Combining base clusters for T1=0.5, T2=0.6 and collection size=100.

Word1	No. of doc. found	Word2	No. of doc. found	No. of doc. in Union	No. of doc. in Intersect	Sim.
hotel	41	thailand hotel	26	41	26	0.634
travel	28	guide	22	33	17	0.515
hotel reservation	9	thailand hotel reservation	5	9	5	0.555
mai	8	chiang mai	7	8	7	0.875
cheap	7	offer cheap	5	7	5	0.714
hotel travel	5	thailand hotel travel	3	5	3	0.600
hotel travel	5	hotel travel guide	3	5	3	0.600
save hotel	4	offer cheap hotel	3	4	3	0.750
low	4	guarantee low	3	4	3	0.750
change	4	stock change thailand	3	4	3	0.750
accommodate guide	3	reservation accommodate	3	3	3	1.000

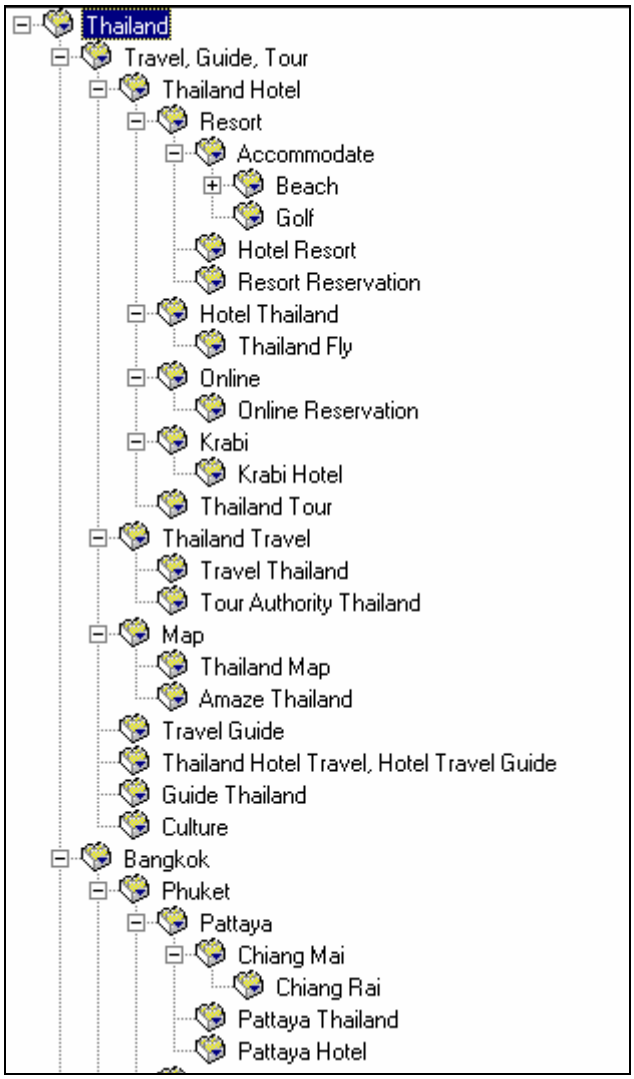


Figure 5.18: Clustering results for T1=0.5 T2=0.6 and collection size=100.

Table 5.21: Combining base clusters for T1=0.5, T2=0.5 and collection size=100.

Word1	No. of doc. found	Word2	No. of doc. found	No. of doc. in Union	No. of doc. in Intersect	Sim.
hotel	41	thailand hotel	26	41	26	0.634
travel	28	guide	22	33	17	0.515
hotel reservation	9	thailand hotel reservation	5	9	5	0.555
mai	8	chiang mai	7	8	7	0.875
cheap	7	offer cheap	5	7	5	0.714
hotel travel	5	thailand hotel travel	3	5	3	0.600
hotel travel	5	hotel travel guide	3	5	3	0.600
save hotel	4	offer cheap hotel	3	4	3	0.750
low	4	guarantee low	3	4	3	0.750
change	4	stock change thailand	3	4	3	0.750
accommodate guide	3	reservation accommodate	3	3	3	1.000

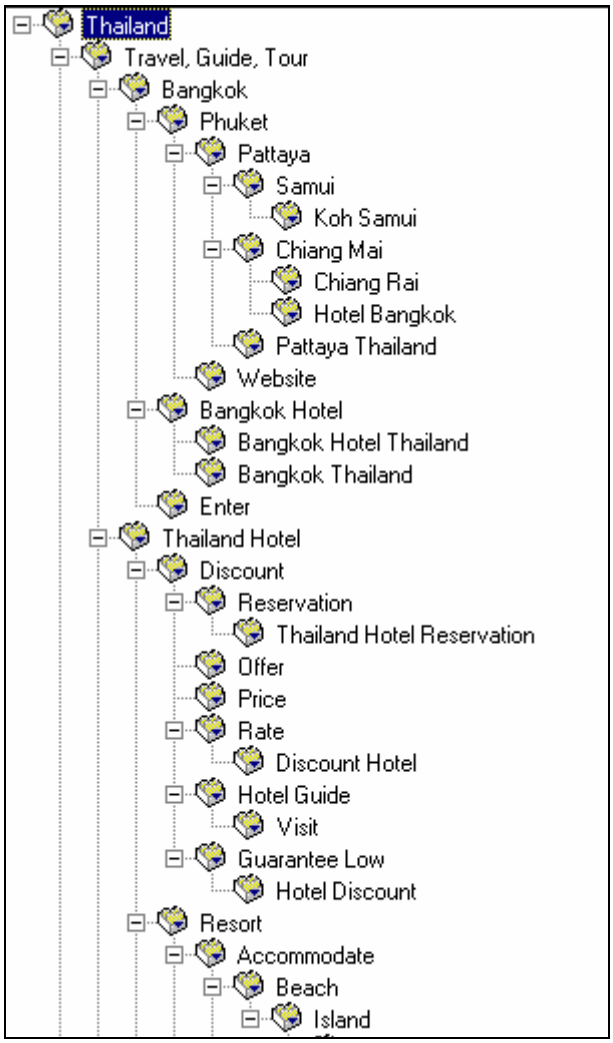


Figure 5.19: Clustering results for T1=0.5 T2=0.5 and collection size=100.

Table 5.22: Combining base clusters for T1=0.8, T2=0.8 and collection size=200.

Word1	No. of doc. found	Word2	No. of doc. found	No. of doc. in Union	No. of doc. in Intersect	Sim.
mai	10	chiang mai	9	10	9	0.900

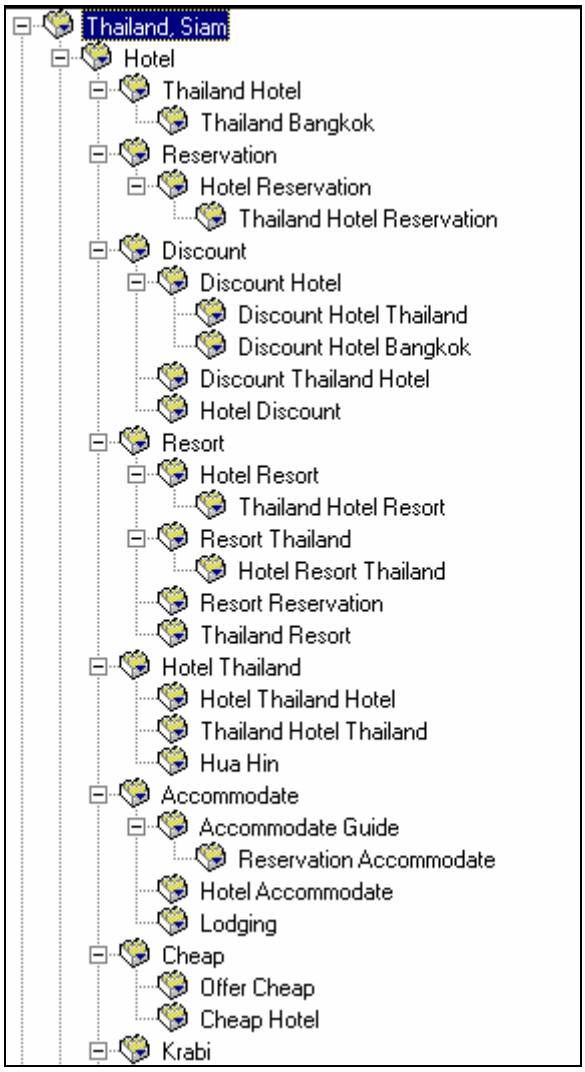


Figure 5.20: Clustering results for T1=0.8, T2=0.8 and collection size=200.

Table 5.23: Combining base clusters for T1=0.7, T2=0.7 and collection size=200.

Word1	No. of doc. found	Word2	No. of doc. found	No. of doc. in Union	No. of doc. in Intersect	Sim.
mai	10	chiang mai	9	10	9	0.900
save	5	save hotel	4	5	4	0.800
authority thailand	5	tour authority thailand	4	5	4	0.800
king	5	king thailand	4	5	4	0.800
hotel guide	8	thailand hotel guide	6	8	6	0.750
engine	4	search engine	3	4	3	0.750
vietnam	4	vietnam hotel	3	4	3	0.750
advice	4	travel advice	3	4	3	0.750
accommodate guide	4	reservation accommodate	3	4	3	0.750
change	4	stock change thailand	3	4	3	0.750

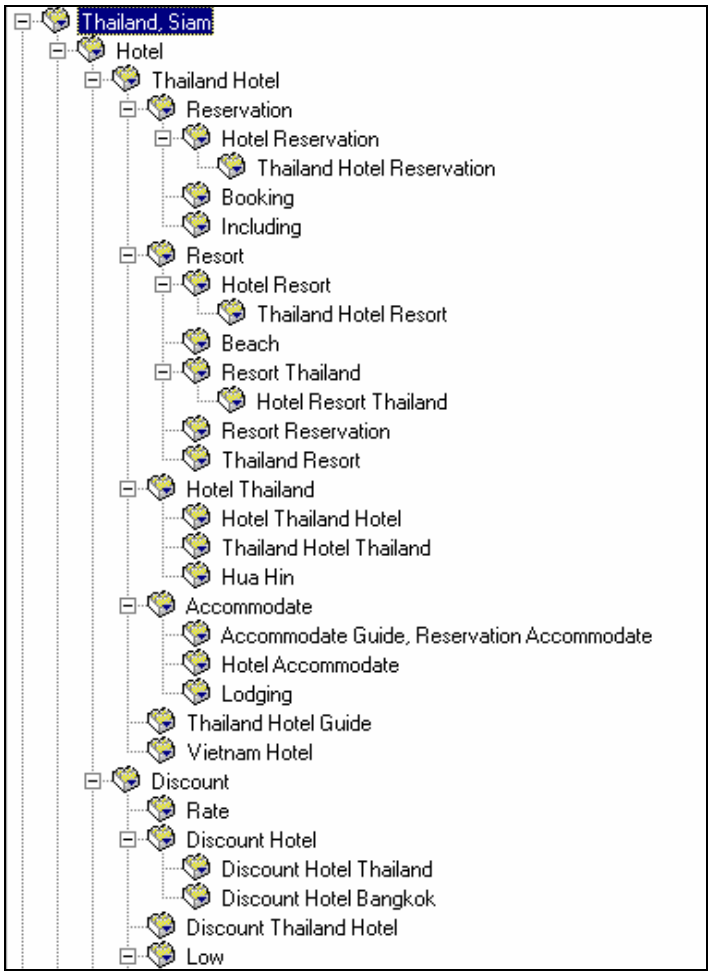


Figure 5.21: Clustering results for T1=0.7, T2=0.7 and collection size=200.

Table 5.24: Combining base clusters for T1=0.6, T2=0.6 and collection size=200.

Word1	No. of doc. found	Word2	No. of doc. found	No. of doc. in Union	No. of doc. in Intersect	Sim.
mai	10	chiang mai	9	10	9	0.900
save	5	save hotel	4	5	4	0.800
authority thailand	5	tour authority thailand	4	5	4	0.800
king	5	king thailand	4	5	4	0.800
hotel guide	8	thailand hotel guide	6	8	6	0.750
engine	4	search engine	3	4	3	0.750
vietnam	4	vietnam hotel	3	4	3	0.750
advice	4	travel advice	3	4	3	0.750
accommodate guide	4	reservation accommodate	3	4	3	0.750
change	4	stock change thailand	3	4	3	0.750
land	9	land smile	6	9	6	0.666
hotel travel	6	thailand hotel travel	4	6	4	0.666
fly	6	thailand fly	4	6	4	0.666
hotel	64	thailand hotel	40	64	40	0.625

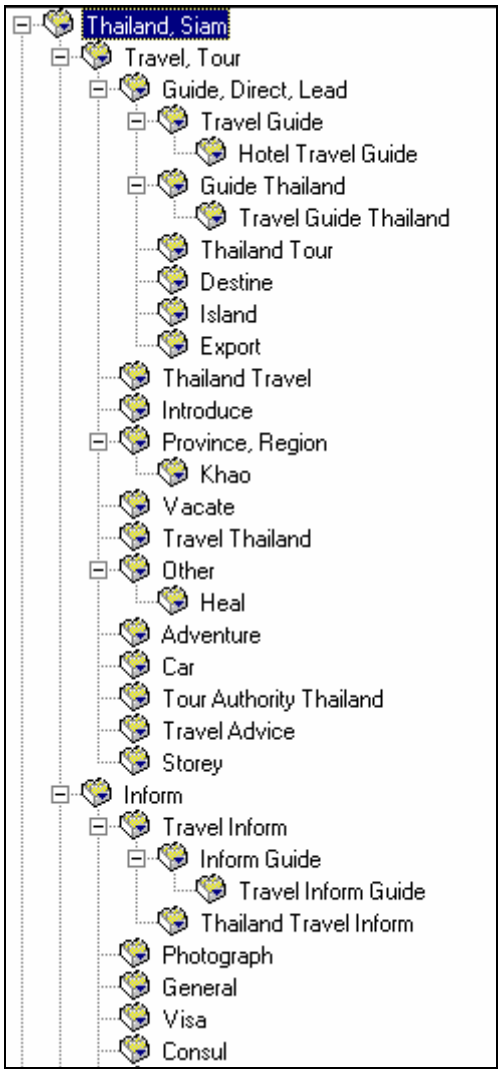


Figure 5.22: Clustering results for T1=0.6, T2=0.6 and collection size=200.

Table 5.25: Combining base clusters for T1=0.8, T2=0.8 and collection size=300.

Word1	No. of doc. found	Word2	No. of doc. found	No. of doc. in Union	No. of doc. in Intersect	Sim.
Chiang mai hotel	3	hotel chiang mai	3	3	3	1.000
mai	14	chiang mai	13	14	13	0.928
koh	13	koh samui	12	13	12	0.923
engine	6	search engine	5	6	5	0.833
amaze	6	amaze thailand	5	6	5	0.833

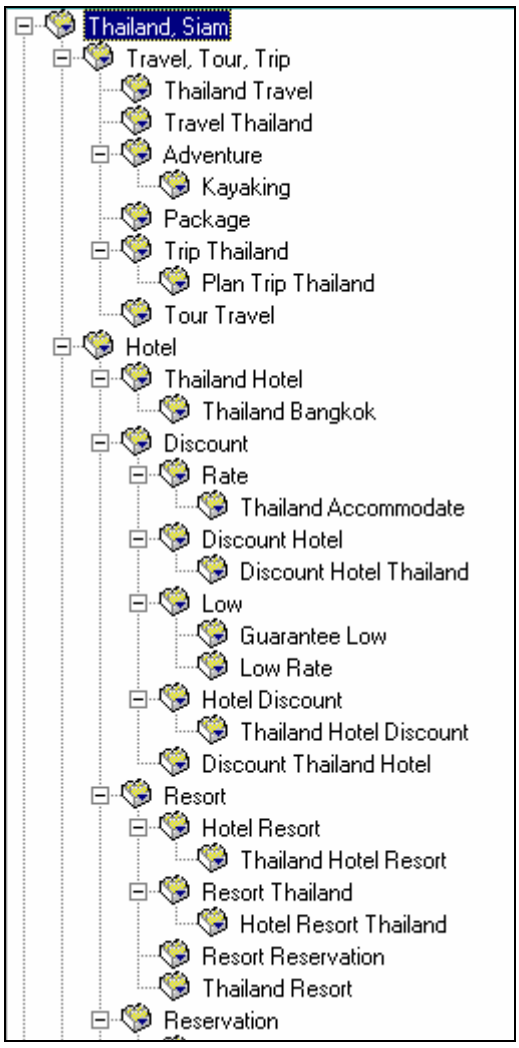


Figure 5.23: Clustering results for T1=0.8, T2=0.8 and collection size=300.

Table 5.26: Combining base clusters for T1=0.7, T2=0.7 and collection size=300.

Word1	No. of doc. found	Word2	No. of doc. found	No. of doc. in Union	No. of doc. in Intersect	Sim.
chiang mai hotel	3	hotel chiang mai	3	3	3	1.000
mai	14	chiang mai	13	14	13	0.928
koh	13	koh samui	12	13	12	0.923
engine	6	search engine	5	6	5	0.833
amaze	6	amaze thailand	5	6	5	0.833
save	5	save hotel	4	5	4	0.800
teach	5	teach england	4	5	4	0.800
authority thailand	5	tour authority thailand	4	5	4	0.800
accommodate guide	5	reservation accommodate	4	5	4	0.800
trip thailand	4	plan trip thailand	3	4	3	0.750
lao	4	myanmar	3	4	3	0.750
krabi hotel	4	discount krabi hotel	3	4	3	0.750
samui hotel thailand	4	samui hotel thailand hotel	3	4	3	0.750
school	4	school thailand	3	4	3	0.750
lao	4	cambodia	3	4	3	0.750
hong	4	hong kong	3	4	3	0.750
king	14	king thailand	10	14	10	0.714
pattaya hotel	7	hotel pattaya	5	7	5	0.714

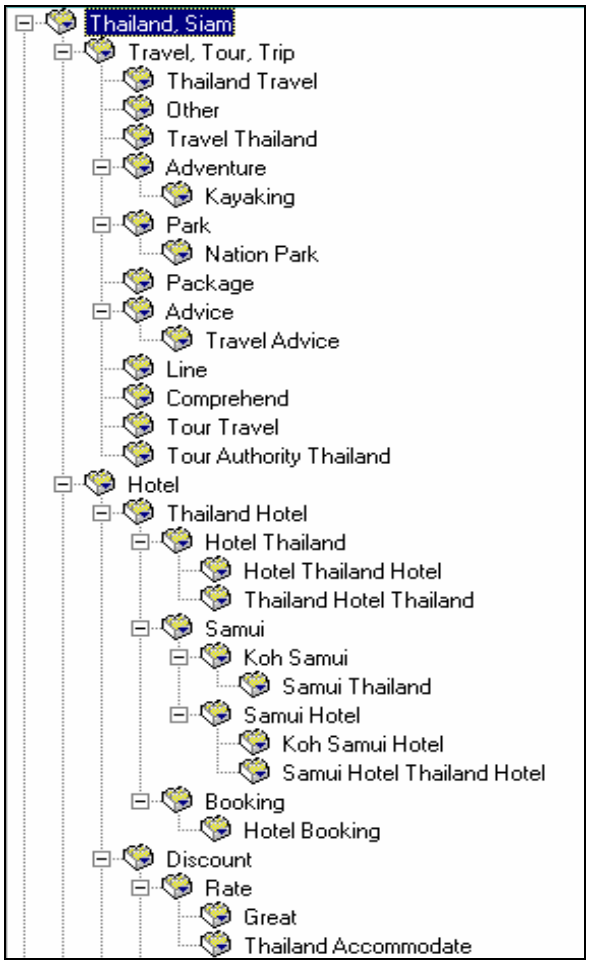


Figure 5.24: Clustering results for T1=0.7, T2=0.7 and collection size=300.

Table 5.27: Combining base clusters for T1=0.6, T2=0.6 and collection size=300.

Word1	No. of doc. found	Word2	No. of doc. found	No. of doc. in Union	No. of doc. in Intersect	Sim.
chiang mai hotel	3	hotel chiang mai	3	3	3	1.000
mai	14	chiang mai	13	14	13	0.928
koh	13	koh samui	12	13	12	0.923
engine	6	search engine	5	6	5	0.833
amaze	6	amaze thailand	5	6	5	0.833
save	5	save hotel	4	5	4	0.800
teach	5	teach england	4	5	4	0.800
authority thailand	5	tour authority thailand	4	5	4	0.800
accommodate guide	5	reservation accommodate	4	5	4	0.800
trip thailand	4	plan trip thailand	3	4	3	0.750
lao	4	myanmar	3	4	3	0.750
krabi hotel	4	discount krabi hotel	3	4	3	0.750
samui hotel thailand	4	samui hotel thailand hotel	3	4	3	0.750
school	4	school thailand	3	4	3	0.750
lao	4	cambodia	3	4	3	0.750
hong	4	hong kong	3	4	3	0.750
king	14	king thailand	10	14	10	0.714
pattaya hotel	7	hotel pattaya	5	7	5	0.714
fly	6	thailand fly	4	6	4	0.667
samui hotel	6	koh samui hotel	4	6	4	0.667
samui hotel	6	samui hotel thailand	4	6	4	0.667
low	6	guarantee low	4	6	4	0.667
hotel guide	6	thailand hotel guide	4	6	4	0.667
hotel	83	thailand hotel	53	83	53	0.638

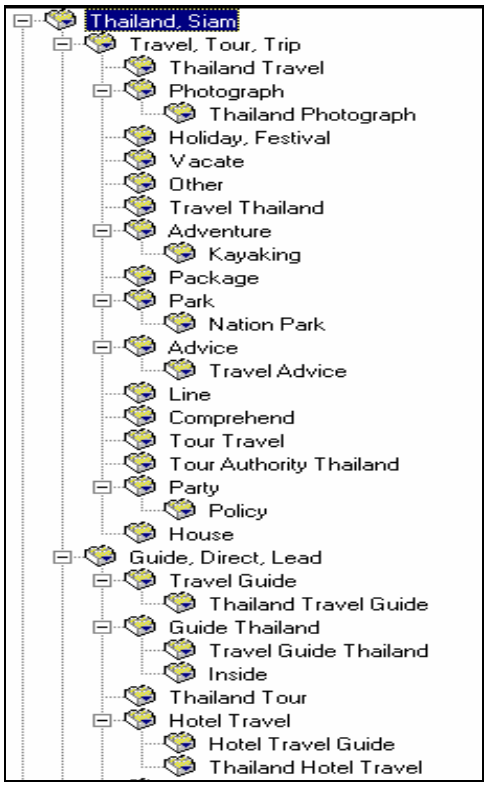


Figure 5.25: Clustering results for T1=0.6, T2=0.6 and collection size=300.

Table 5.28: Combining base clusters for T1=0.8, T2=0.8 and collection size=500.

Word1	No. of doc. found	Word2	No. of doc. found	No. of doc. in Union	No. of doc. in Intersect	Sim.
hotel chiang mai	3	chiang mai hotel	3	3	3	1.000
mai	18	chiang mai	17	18	17	0.944
engine	8	search engine	7	8	7	0.875
similar	8	similar page	7	8	7	0.875
koh	15	koh samui	13	15	13	0.866
dive	6	scuba diving	5	6	5	0.833
planet	6	lone planet	5	6	5	0.833

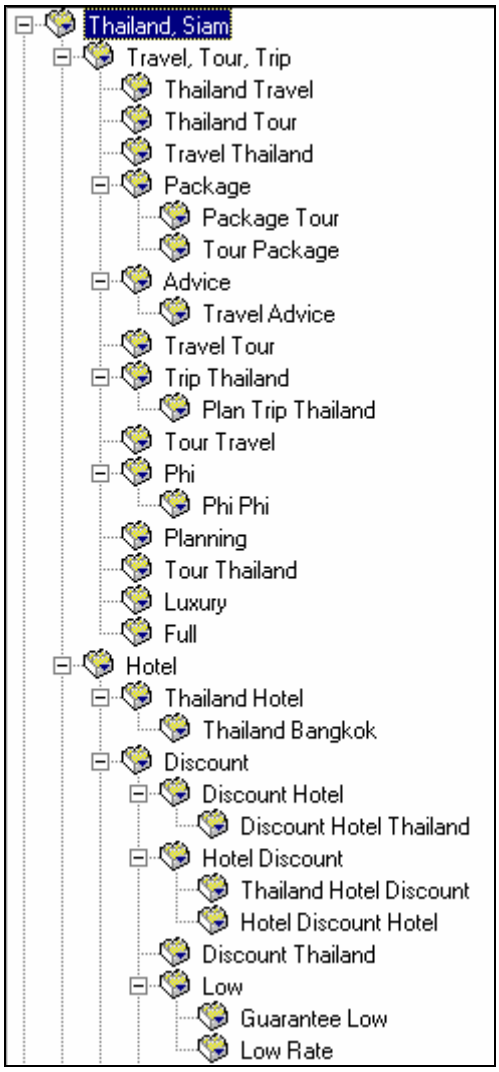


Figure 5.26: Clustering results for T1=0.8, T2=0.8 and collection size=500.

Table 5.29: Combining base clusters for T1=0.7, T2=0.7 and collection size=500.

Word1	No. of doc. found	Word2	No. of doc. found	No. of doc. in Union	No. of doc. in Intersect	Sim.
hotel chiang mai	3	chiang mai hotel	3	3	3	1.000
mai	18	chiang mai	17	18	17	0.944
engine	8	search engine	7	8	7	0.875
similar	8	similar page	7	8	7	0.875
koh	15	koh samui	13	15	13	0.866
dive	6	scuba diving	5	6	5	0.833
planet	6	lone planet	5	6	5	0.833
discount thailand	10	discount thailand hotel	8	10	8	0.800
authority thailand	5	tour authority thailand	4	5	4	0.800
stock	5	stock change thailand	4	5	4	0.800
hin	5	hua hin	4	5	4	0.800
hotel online	5	thailand hotel online	4	5	4	0.800
southeast	9	southeast asia	7	9	7	0.777
know siam	4	thailand southeast asia country european power	3	4	3	0.750
today	4	hotel today	3	4	3	0.750
thailand embassy	4	royal thailand embassy	3	4	3	0.750
yellow	4	yellow page	3	4	3	0.750
thailand sea kayaking	4	thailand sea kayaking adventure	3	4	3	0.750
thailand sea kayaking	4	adventure eco tour	3	4	3	0.750
thailand search	4	thailand search engine	3	4	3	0.750
affair	4	foreign affair	3	4	3	0.750
samui hotel thailand	4	samui hotel thailand hotel	3	4	3	0.750
muang	4	muang thailand	3	4	3	0.750
thailand hotel thailand	4	thailand hotel thailand hotel	3	4	3	0.750
high	4	high quality	3	4	3	0.750
hong	4	hong kong	3	4	3	0.750
cia	4	world factbook	3	4	3	0.750
phi	4	phi phi	3	4	3	0.750
phuket hotel resort	4	phuket hotel resort thailand	3	4	3	0.750
save hotel	4	offer cheap hotel	3	4	3	0.750
north	7	north thailand	5	7	5	0.714
east asia	7	south east asia	5	7	5	0.714

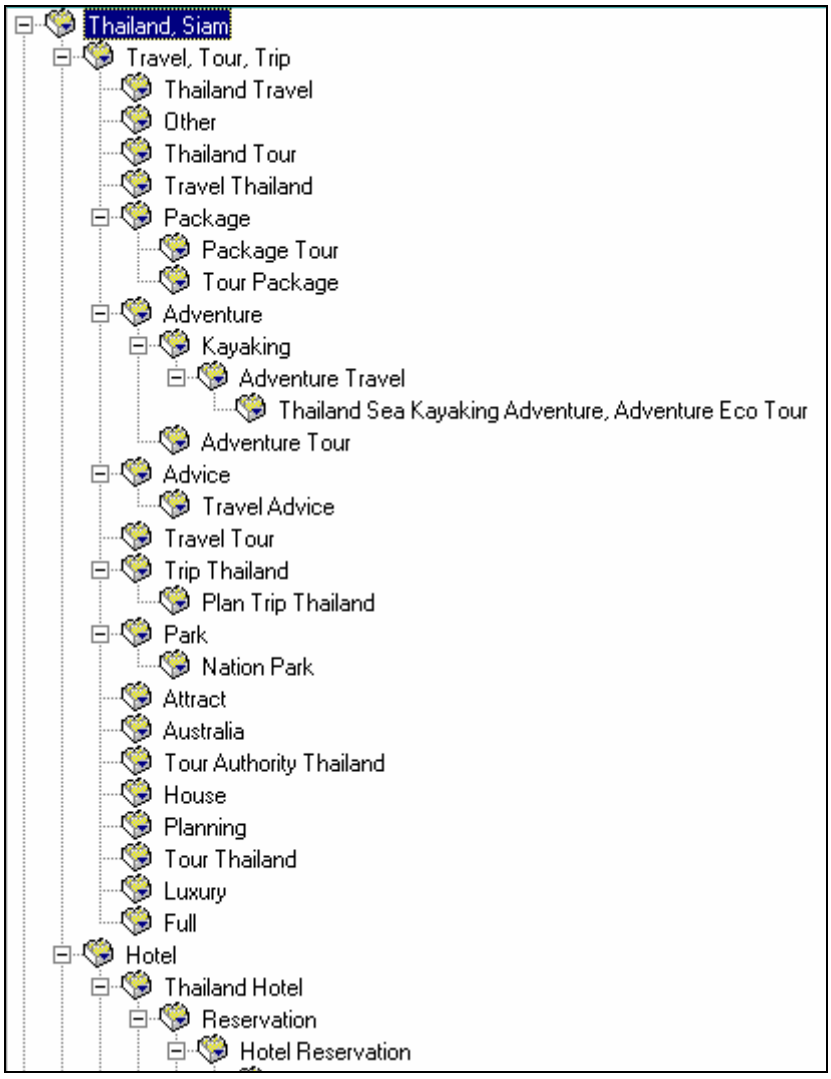


Figure 5.27: Clustering results for T1=0.7, T2=0.7 and collection size=500.

Table 5.30: Combining base clusters for T1=0.6, T2=0.6 and collection size=500.

Word1	No. of doc. found	Word2	No. of doc. found	No. of doc. in Union	No. of doc. in Intersect	Sim.
hotel chiang mai	3	chiang mai hotel	3	3	3	1.000
mai	18	chiang mai	17	18	17	0.944
engine	8	search engine	7	8	7	0.875
similar	8	similar page	7	8	7	0.875
koh	15	koh samui	13	15	13	0.866
dive	6	scuba diving	5	6	5	0.833
planet	6	lone planet	5	6	5	0.833
discount thailand	10	discount thailand hotel	8	10	8	0.800
authority thailand	5	tour authority thailand	4	5	4	0.800
stock	5	stock change thailand	4	5	4	0.800
hin	5	hua hin	4	5	4	0.800
hotel online	5	thailand hotel online	4	5	4	0.800
southeast	9	southeast asia	7	9	7	0.777
know siam	4	thailand southeast asia country european power	3	4	3	0.750
today	4	hotel today	3	4	3	0.750
thailand embassy	4	royal thailand embassy	3	4	3	0.750
yellow	4	yellow page	3	4	3	0.750
thailand sea kayaking	4	thailand sea kayaking adventure	3	4	3	0.750
thailand sea kayaking	4	adventure eco tour	3	4	3	0.750
thailand search	4	thailand search engine	3	4	3	0.750
affair	4	foreign affair	3	4	3	0.750
samui hotel thailand	4	samui hotel thailand hotel	3	4	3	0.750
muang	4	muang thailand	3	4	3	0.750
thailand hotel thailand	4	thailand hotel thailand hotel	3	4	3	0.750
high	4	high quality	3	4	3	0.750
hong	4	hong kong	3	4	3	0.750
cia	4	world factbook	3	4	3	0.750
phi	4	phi phi	3	4	3	0.750
phuket hotel resort	4	phuket hotel resort thailand	3	4	3	0.750
save hotel	4	offer cheap hotel	3	4	3	0.750
north	7	north thailand	5	7	5	0.714
east asia	7	south east asia	5	7	5	0.714
hotel guide	10	thailand hotel guide	7	10	7	0.700
king	23	king thailand	16	23	16	0.695

hotel discount	12	thailand hotel discount	8	12	8	0.666
thailand hotel resort	9	thailand hotel resort reservation	6	9	6	0.666
amaze	9	amaze thailand	6	9	6	0.666
diving	9	dive	6	9	6	0.666
samui hotel	6	koh samui hotel	4	6	4	0.666
samui hotel	6	samui hotel thailand	4	6	4	0.666
tel	6	thailand tel	4	6	4	0.666
sea	6	thailand sea kayaking	4	6	4	0.666
trip Thailand	6	plan trip thailand	4	6	4	0.666
profile	6	country profile	4	6	4	0.666
human	6	human right	4	6	4	0.666
reservation accommodate	6	accommodate guide	4	6	4	0.666
hotel	113	thailand hotel	75	113	75	0.663
teach	8	teach england	5	8	5	0.625
real	8	real estate	5	8	5	0.625
low	8	guarantee low	5	8	5	0.625
pattaya hotel	8	hotel pattaya	5	8	5	0.625

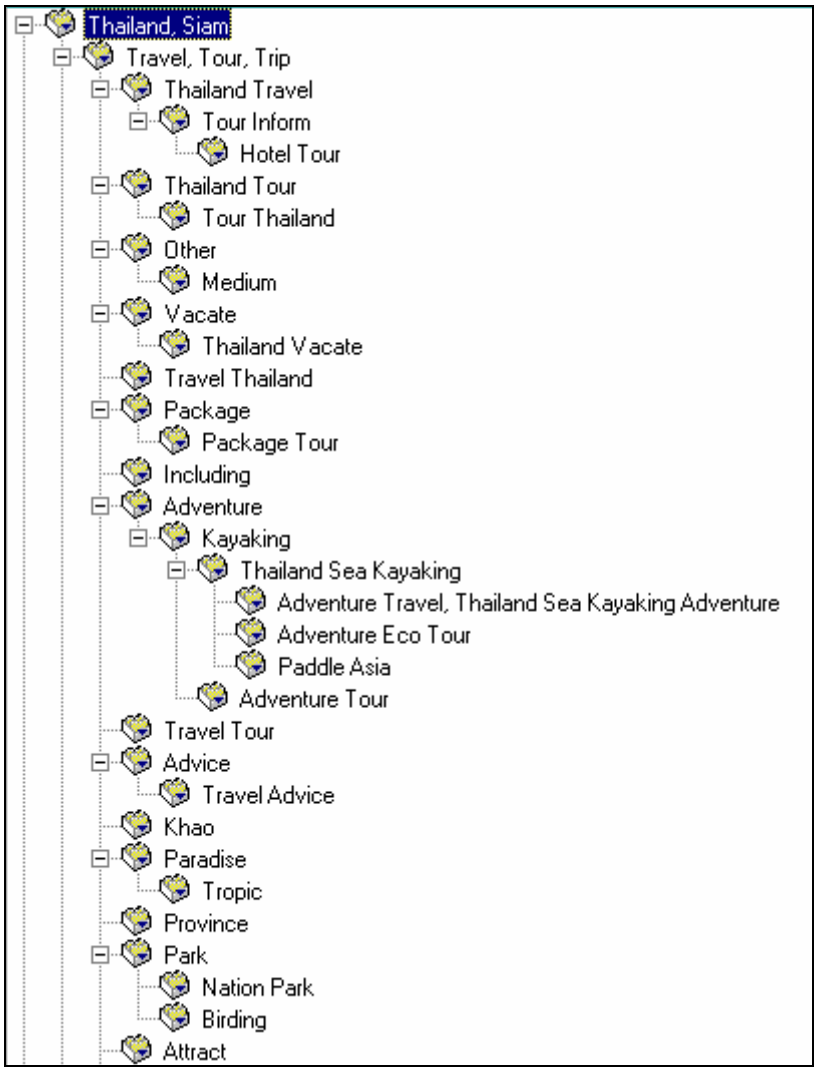


Figure 5.28: Clustering results for T1=0.6, T2=0.6 and collection size=500.

5.4 Experiments for SAC processing Time

This type of experiments concentrates on the processing time comparing between the two techniques: SAC and STC. Because the parameters, T1 and T2, have minor or no effect on the processing time, we will conduct the experiments of SAC upon various sizes of document collection. We formulate 3 sets of test data which each set consists of 10 document collections by collecting documents from Google search engine in the same way as previous type of experiments in section 5.3. The document collections have different sizes ranging from 100 to 1000 documents per collection. The parameters T1 and T2 are fixed at 0.8 and 0.8 respectively. We also conduct the experiments on STC which is developed and run under the same environment as SAC. The first set of document collections is obtained by using a query keyword “Thailand” and its summary is shown in Table 5.31. The processing times of SAC and STC upon this set of collections are compared in Figure 5.29.

Table 5.31: A set of document collections obtained by using query keyword “Thailand”.

Collection no.	Number of Documents	Collection Size (Bytes)
1	100	94,563
2	200	188,023
3	300	282,788
4	400	370,973
5	500	460,447
6	600	548,887
7	700	637,595
8	800	723,473
9	900	811,546
10	1,000	838,894

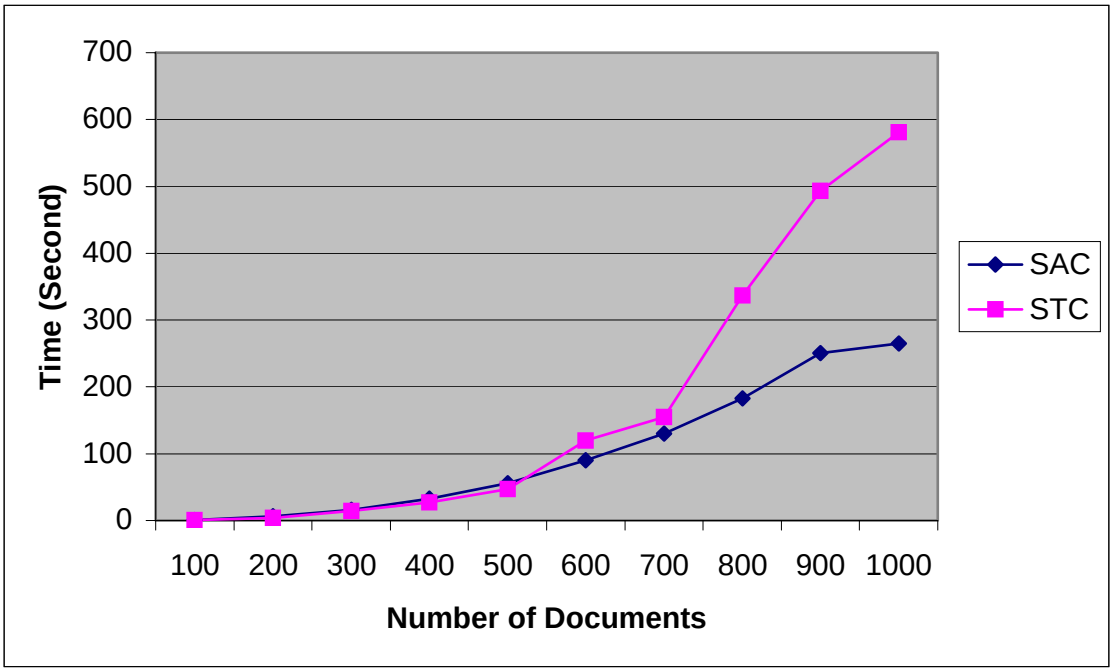


Figure 5.29: The comparisons of processing times between SAC and STC for collections with keyword "Thailand".

We see from the experimental results that for small collections, size 100-500 documents, SAC and STC performance are almost the same. However, for larger collections, processing times of STC grows more rapidly than that of SAC. For 1000-document collection, STC takes nearly 600 seconds while SAC takes less than 300 seconds.

We perform two more sets of experiments similar to the first one but using different keywords of "Sport" and "Entertainment". The sizes of collections tested and the comparison of processing times are shown in Table 5.32 – 5.33 and Figure 5.30 – 5.31.

Table 5.32: A set of document collections obtained by using query keyword “Sport”.

Collection no.	Number of Documents	Sizes of Data (Bytes)
1	100	71,316
2	200	144,459
3	300	218,538
4	400	296,029
5	500	372,689
6	600	450,378
7	700	528,658
8	800	608,778
9	900	667,053

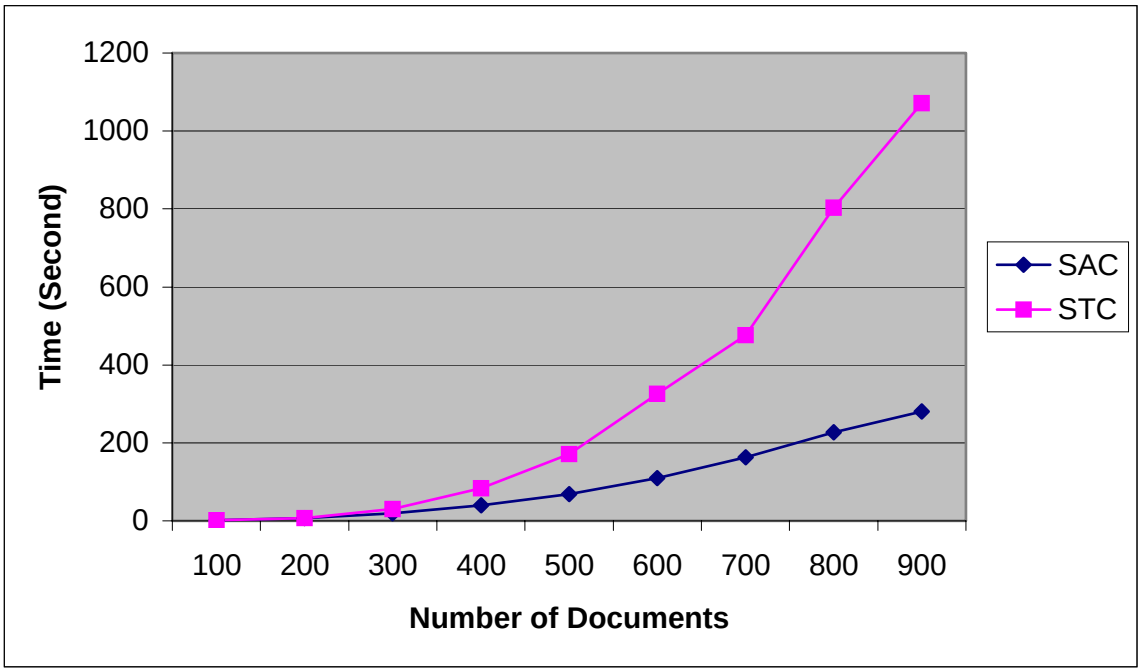


Figure 5.30: The comparisons of processing times between SAC and STC for collections with keyword “Sport”.

Table 5.33: A set of document collections obtained by using query keyword "Entertainment".

Collection no.	Number of Documents	Sizes of Data (Bytes)
1	100	71,641
2	200	143,714
3	300	214,809
4	400	286,835
5	500	361,222
6	600	436,081
7	700	512,976
8	800	587,060
9	900	661,458

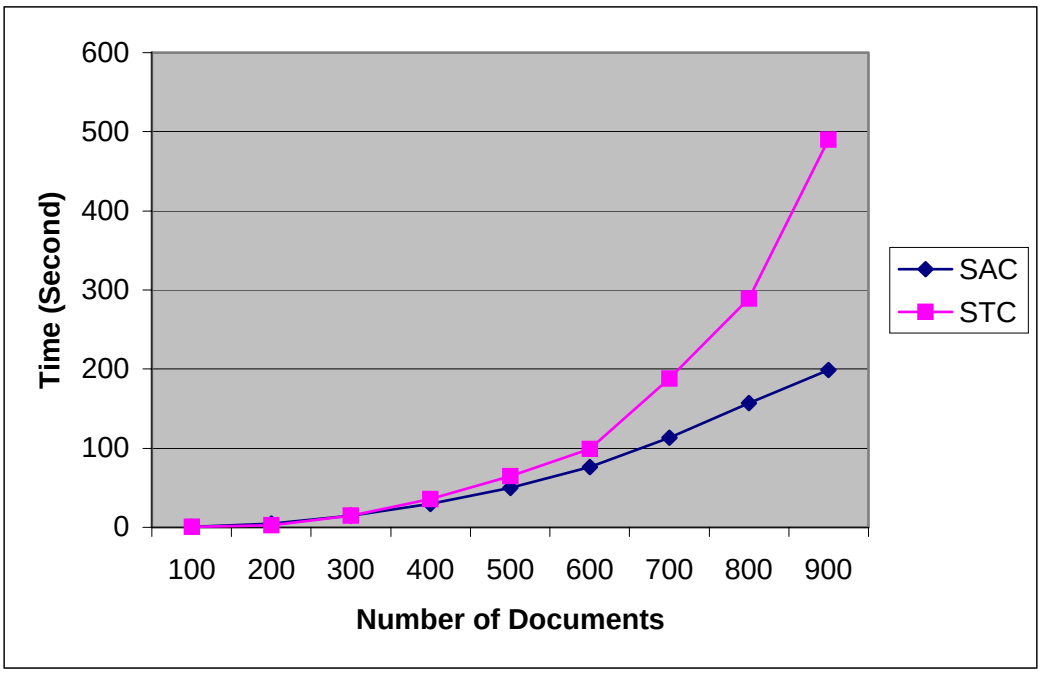


Figure 5.31: The comparisons of processing times between SAC and STC for collections with keyword "Entertainment".

We also do further experiments on other data types beside the data received from Google. News information from many news agencies, eg. CNN and Reuters are selected for our experiments. Normally, these news information available on the Internet can be separated into 2 sections: Headline and Story. In the experiments, we use the Story part to perform document clustering to compare the processing times between STC and SAC techniques. From the experimental results, SAC still outperforms STC for all three sets of test data. The histograms shown in Figure 5.32 compare processing times between the two clustering techniques.

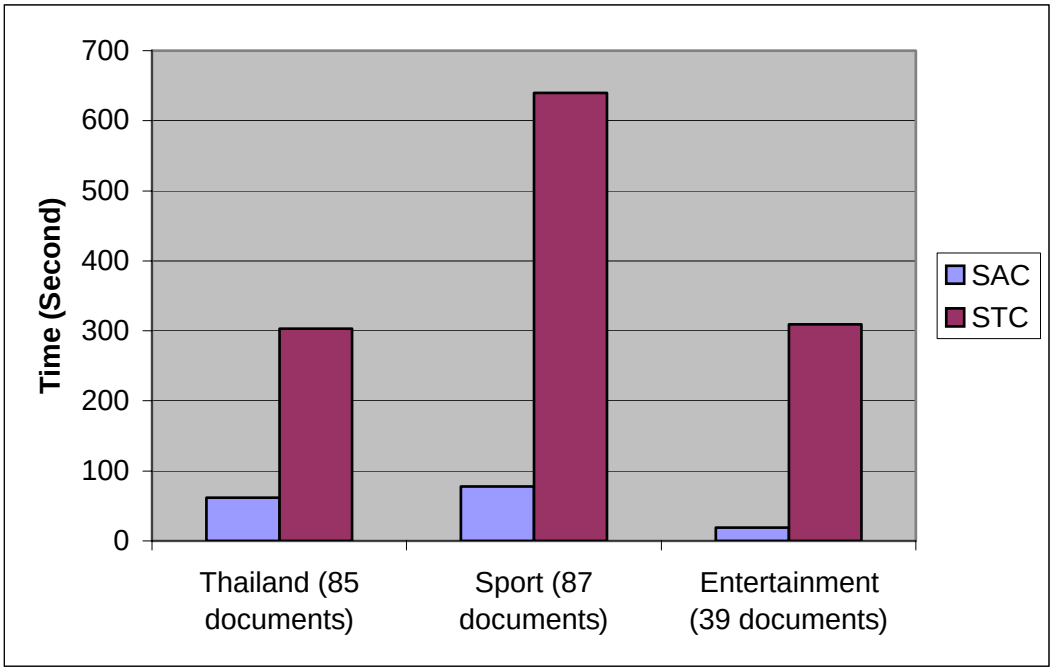


Figure 5.32: The comparison of processing times between SAC and STC on news collections.

Next, we compare the performance of SAC in term of precision with its two variants, *SAC-no-overlap* which forces the results into only one partition and *SAC-no-phrases* which uses only single word phrase. Figure 5.33 shows the precision performance of the three variants obtained from the experiments upon 10 selected collections.

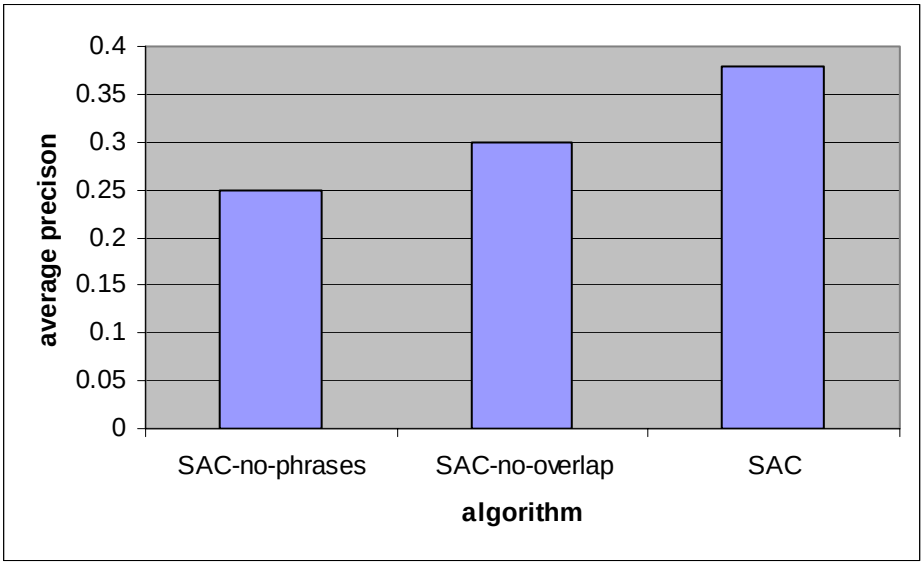


Figure 5.33: The average precision of the 10 selected collections using variants of SAC: *SAC-no-overlap* and *SAC-no-phrases*.

Finally, we compare the precision performance of SAC with some other clustering techniques currently in use, STC, Single-Pass and K-Means. Among these techniques, SAC and STC give the best performance. Figure 5.34 shows the average precision provided by the four clustering techniques.

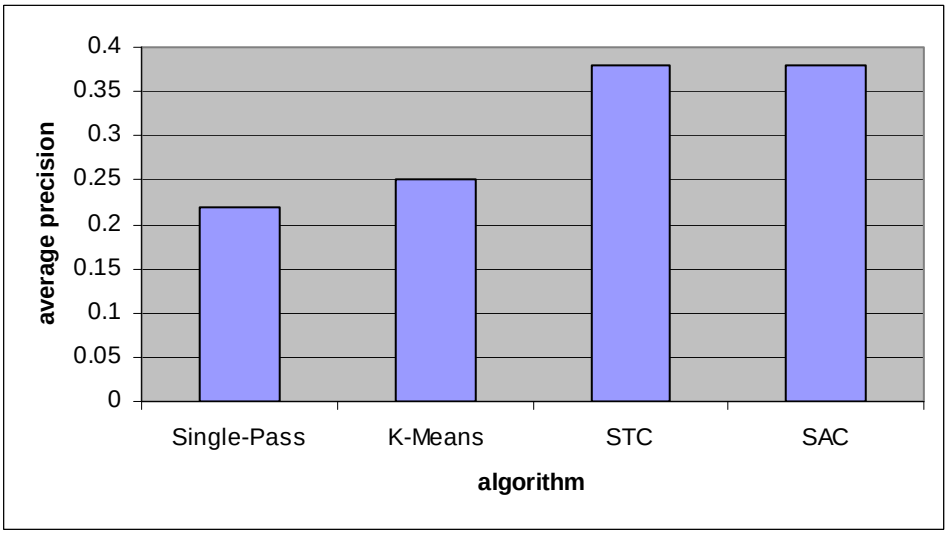


Figure 5.34: Comparison of the average precision among 4 clustering algorithms, Single-Pass, K-Means, STC and SAC.

5.5 Comparison of the memory usage between STC and SAC

By the nature of the tree structure, we obviously see that a lot of memory space is used to store many pointers, e.g. parent pointers, left-right pointers, sibling pointers, etc. This causes so much memory overhead. However, this overhead can be avoided in SAC because such pointers are not required in the array structure. We show the actual source programs used to run STC and SAC in Figure 5.35 and Figure 5.36, respectively.

```
struct
{
        char sWord[30];         // node phrase
        int   nParent;          // node parent
        int* nListPosChild;     // child list of node that each node has at least 2 children
} suffix_tree;
```

Figure 5.35: STC structure in the program.

```
struct
{
        char sWord[30];         // node phrase
        int   nLCP              // LCP value of node
} suffix_array;
```

Figure 5.36: SAC structure in the program.

In STC program, a node of phrase requires 4 bytes for parent pointer, about 8 bytes or more for child pointer, and 30 bytes for a phrase itself. On the other hand in SAC program, a node requires 4 bytes for LCP value and 30 bytes to store a phrase itself.

In our prototype of SAC, we found that about 19% in reduction of memory space for actual implementation of each node.

5.6 Summary

This chapter presented the experimental results conducted on the prototype of SAC with various types of test data. The results showed that SAC provided the clustering precision level similar to the clustering technique using suffix tree (STC). However, SAC outperformed STC in terms of response time and memory consumption. SAC also performed better than Single-Pass and K-Means clustering algorithms in term of clustering precision. In the next chapter, we will discuss SAC on some interesting issues and conclusion will be summarized.

CHAPTER VI
DISCUSSION AND CONCLUSION

6.1 Discussion

The Suffix Tree Clustering (STC) is not a new technique but used for decades in clustering applications. From the study, it reveals that this technique is rather difficult for implementation and consumes large memory space. Thus, we proposed an alternative technique to resolve the above troubles. Many algorithms in the area of string matching are examined and according to our study, the probably more suitable technique of Suffix Array Clustering (SAC) is introduced.

The STC employs suffix tree algorithm for performing string matching in each document to compute similarity values of documents in a collection. However, due to its high complexity, the suffix tree algorithm is replaced by suffix array algorithm for string matching. Later on, this suffix array algorithm is referred to as Suffix Array Clustering (SAC). According to our study, both techniques return the same results for document clustering. However, we found some differences in the performance of clustering. Firstly, obviously, even array has the same complexity with tree, its structure is simpler. This results in easier implementation for SAC. Secondly, array needs few pointers to maintain its structure that contrary to the tree structure, which contain many of pointers to link to. This result in less memory space required for SAC in performing clustering. Thirdly, we found that STC is faster than SAC for small document collection but SAC outperforms STC for large document collection. Referring to figure 5.31 in Chapter 5, SAC performs better than STC for over 400 documents. The small document collection is referred to the document collection containing not many different words such that suffix tree and its auxiliary files can be

wholly created in computer memory. On the contrary, the large document collection is referred to the document collection containing many different words such that the whole suffix tree and its auxiliary files cannot be fit in computer main memory currently available. The reason of degrading in time performance is that, when document collection becomes larger, both tree and array become larger too. However, the tree, with more space overhead, grows faster than array. This causes more difficulties and more time in maintaining their structures when the structures go beyond the memory capacity. However, it is more probably to occur with STC rather than SAC when we have to deal with a large document collection.

In summary, we would like to emphasize that the two techniques have different cost/benefits. If the search engine providers focus more on the speed of execution, they may prefer to use the STC technique and bare additional cost of the memory extension. Otherwise, the providers may opt for the SAC technique that consumes less memory especially for massive document sizes.

6.2 Conclusions

After our analysis and experimental results evaluation, we come to the conclusions listed as follows.

1. STC and SAC produce exactly the same results of clustering.
2. STC is faster than SAC when applying to small document collections.
3. SAC is faster than STC when applying to large document collections.
4. SAC requires less memory space than STC in clustering process.
5. SAC has less complexity and easier to implement than STC. Hence, SAC is more practical than STC is in document clustering.

6.3 Future Work

The following researches should be further surveyed and explored to increase the quality of document clustering.

6.3.1 Polysemy and Synonymy

Besides selecting the technique of document clustering to get proper results, one of the most concerns found in document clustering method development is interpretation of the meaning of words which includes:

- The synonymy (different words have similar meanings)
- The polysemy (same words have different meanings)

These two characteristics are found in most languages. The synonymy has been considered partly in this thesis via a simple thesaurus (synonym words [22] are shown in Appendix A). However, the polysemy has not put into consideration yet. If the two characteristics have been added into a clustering technique, it can make the document clustering method provides more reliable results.

6.3.2 Data Compression Application

Beside applying STC or SAC directly to document clustering, they are possibly applied for data compression. Because clustering groups of documents having common properties provides the feasibility to do data compaction. However, the application of clustering to this field needs more extensively studying.

REFERENCES

1. Jain, A. K. and Dubes, R. C. 1988. Algorithms for Clustering Data, Prentice Hall.

2. Zamir, O. and Etzioni, O. 1998 Web Document Clustering: A Feasibility Demonstration In Proceedings of ACM/SIGIR.

3. Salton, G. and Buckley, C. 1988. Term Weighting Approaches in Automatic Text Retrieval, Information Processing and Management.

4. Botafogo, R. A. 1993. Cluster Analysis for HyperText System, ACM- SIGIR'93, Pitsburgh.

5. Van Rijsbergen, C. J. 1979. Information Retrieval, Department of Computer Science, University of Galasgow, Second edition.

6. Ukkonen, E. 1995. On-line Construction of Suffix Trees.

7. Anil, K. J. and Richard, C. D. 1988. Algorithms for Clustering Data. Prentice Hall.

8. Vance, F. 1994. Clustering and the Continuous K-Means Algorithm.

9. Fox, C. 1992. Lexical analysis and stoplists. In Information Retrieval - Data Structures & Algorithms.

10. Guihong, C. and Peter, B. 2003. Suffix Tree Clustering on Post-retrieval Document of The University of Queensland.

11. Wu, S. and Manber U. 1992. A fast text searching allowing errors.

12. Gusfield, D. 1997. Algorithms on Strings, Trees and Sequences. Cambridge University Press, New York.

13. Alfread, V. A. and Jeffrey, D. U. 1983. Data Structures and Algorithms.

14. Porter, M. F. 1980. An Algorithm For Suffix Stripping.

15. http://www.lycos.com/

16. http://www.altavista.com/

17. http://www.yahoo.com/

18. http://www.google.com/

19. Tran Ngoc Thanh. 1999. Applying Document Clustering to HyperText Document on the Internet.

20. http://citeseer.ist.psu.edu/context/50862/0

21. Manber, U. and Myers, E. 1990. Suffix arrays: A new method for on-line string searches. In Proceedings of the first Annual ACM-SIAM Symposium on Discrete Algorithms.

22. ThaiSoftware Dictionary version 3.0

APPENDIX

APPENDIX A

A.1 Stopword List

a	did	inc.	off	those
about	didn	indeed	often	through
above	didn't	instead	on	throughout
according	do	into	one	thru
across	does	is	only	till
actually	doesn	isn	onto	to
adj	doesn't	isn't	or	together
after	doing	it	others	toward
afterwards	don	its	otherwise	towards
again	done	itself	our	two
against	don't	jan	ours	under
ago	down	january	ourselves	unless
all	during	jul	out	unlike
almost	each	july	over	unlikely
alone	eg	jun	own	until
along	else	june	per	up
also	elsewhere	just	perhaps	upon
although	end	last	put	us
always	ending	later	putting	use
am	ends	latterly	rather	used
among	enough	least	re	using
amonges	etc	left	recent	ve
an	even	less	recently	very
and	ever	let	same	via
another	every	like	saw	want
any	everyone	likely	say	was
anybody	everything	ll	see	wasn
anyhow	everywhere	ltd	seem	we

anyone	except	made	seemed	well
anything	feb	make	seeming	went
anyway	february	makes	seems	were
anywhere	few	many	seen	weren
apr	for	mar	sep	weren't
april	from	march	september	what
are	front	may	several	whatever
aren	further	may	shall	what's
aren't	get	may	she	when
around	getting	maybe	should	whence
as	go	me	shouldn	whenever
at	goes	meantime	shouldn't	where
aug	going	meanwhile	since	whereafter
august	gone	might	so	whereas
away	got	mine	some	whereby
back	gotten	miss	somebody	wherein
be	had	month	somehow	whereupon
became	has	more	someone	wherever
because	hasn	moreover	something	whether
become	hasn't	most	sometime	which
becomes	have	mostly	sometimes	while
been	haven	mr	somewhere	who
before	haven't	mrs	stand	whoever
beforehand	having	much	still	whole
begin	he	must	such	whom
behind	hence	my	sure	whomever
being	her	myself	take	whose
below	here	namely	taking	why
beside	hereby	need	than	will
besides	herein	never	that	with
between	hereupon	next	the	within
both	hers	no	their	without
but	herself	nobody	them	won't
by	him	none	themselves	would
came	himself	nonetheless	then	wouldn
can	his	noone	thence	wouldn't
cannot	hour	nor	there	wouldn't

can't	hours	not	thereafter	year
co	how	nothing	thereby	yes
come	however	nov	therefore	yet
could	i	november	therein	you
couldn	i.e.	now	thereupon	your
couldn't	ie	nowhere	these	yours
day	If	oct	they	yourself
dec	In	october	this	yourselves
december	inc	of	those	

A.2 The table of word passed the Stemming process and Synonym process from the titles that are the results from searching with “Thailand” in 1,000 titles.

From	To	From	To
academic	academy	issues	Issue
academics	academic	items	item
accessible	access	itineraries	itinerant
accomidations	accomidation	itinerary	itinerant
accommodating	accommodate	itis	iti
accommodation	accommodate	iws	iw
accommodations	accommodate	jacobs	jacob
accountable	a/c	james	jame
accuracy	accurate	japanese	japan
accused	accuse	jeeps	jeep
achievements	achieve	jewelry	jewel
acids	acid	jobs	job
acknowledgments	acknowledge	joined	join
actions	act	joint	join
active	act	journalists	journal
activities	act	journeys	journey
adaptive	adapt	judicial	judiciary
addition	add	jungles	jungle
additional	add	kept	keep
additions	add	keyboards	keyboard
addressed	address	keywords	keyword
administrative	administer	kilometres	kilometer
administrator	administer	kinds	kind
admissions	admit	kingdom	king
adoption	adopt	kings	king
adoptive	adopt	kitefliers	kiteflier
adopts	adopt	knowledge	know
ads	ad	known	know
adults	adult	kss	ks
advanced	advance	laboratory	labor
advances	advance	ladies	lady

adventures	adventure	languages	language
advertisement	advertise	laos	lao
advertisings	advertising	las	la
advised	advice	latest	late
advises	advice	lay	lie
aeon	eon	leader	lead
aeonts	aeont	leaders	lead
affairs	affair	leadership	lead
affects	affect	leading	lead
afs	af	leagues	league
agencies	agent	learning	learn
agency	agent	led	lead
agents	agent	les	le
aids	aid	lesbians	lesbian
aims	aim	letters	letter
airlines	airline	lexus	lexu
airports	airport	license	licence
airways	airway	life	live
airy	air	lifts	lift
alcoholics	alcohol	limited	limit
aldiss	aldis	linked	link
alerts	alert	listed	list
alleged	allege	listings	listing
alliance	ally	lists	list
alternative	alternate	liveaboards	liveaboard
amazing	amaze	lived	liv
american	america	living	live
americas	america	locals	local
analysis	analyse	located	local
analytical	analyse	location	local
ancestry	ancestor	locations	local
angeles	angele	lonely	lone
animals	animal	looks	look
annotated	annotate	los	lo
annually	annual	loses	lose
answers	answer	lots	lot
antarctica	antarctic	lowest	low

anthems	anthem	luxurious	luxury
apartments	apartment	macaques	macaque
apologies	apology	magazines	magazine
appearance	appear	maildrops	maildrop
appliances	apply	maintained	maintain
applicable	apply	majestic	majesty
application	apply	malaysia	malaya
appointment	appoint	malaysians	malaya
appointments	appoint	maldives	maldive
appoints	appoint	manacles	manacle
approval	approve	management	manage
approximately	approximate	manipulation	manipulate
approximation	approximate	manufactures	manufacture
archaeological	archaeology	maps	map
architectural	architect	markets	market
architecture	architect	marks	mark
areas	area	marriage	marry
aromatics	aromatic	materials	matter
arrangements	arrange	mbps	mbp
arrived	arrive	means	mean
articles	article	med	m
artist	art	medals	medal
artistic	art	media	medium
artists	art	medical	medicine
arts	art	medicines	medicine
asian	asia	meditation	meditate
aspects	aspect	meeting	meet
assembly	assemble	meetings	meet
assessment	assess	meets	meet
assistance	assist	members	member
assisted	assist	membership	member
associated	associate	men	man
association	associate	mentioned	mention
assortment	assort	mermaids	mermaid
assumption	assume	meteorological	meteorology
attitudes	attitude	mice	mouse
attorneys	attorney	microelectronics	microelectronic

attraction	attract	migration	migrate
attractions	attract	migrations	migrate
attracts	attract	ministerial	minister
auctions	auction	ministry	minister
australian	australia	minorities	minor
australians	australia	mints	mint
autos	auto	minutes	minute
available	avail	models	model
avis	avi	modified	modifi
awarded	award	monarchy	monarch
awards	award	monitor	monition
bahamas	bahama	monitors	monition
baker	bake	monkeys	monkey
bands	band	monuments	monument
banking	bank	motels	motel
bars	bar	motor	move
based	base	motors	move
basic	base	mountainous	mountain
beaches	beache	moved	move
bears	bear	movement	move
beautiful	beauty	movies	movy
benefits	benefit	moving	move
bicycles	bicycle	multicoloured	multicolour
bidding	bid	museums	museum
biological	biology	names	name
bit	bite	national	nation
boats	boat	nations	nation
bookings	booking	natural	nature
bookmarks	bookmark	naturetrails	naturetrail
books	book	nearly	near
bookshops	bookshop	necked	neck
borders	border	neighbouring	neighbor
bound	bind	netherlands	netherland
brands	brand	newcomers	newcomer
brides	bride	newspapers	newspaper
briefings	briefing	noodles	noodle
brinks	brink	northern	north

british	britain	notes	note
brotherhood	brother	novels	novel
browsers	browser	numbers	number
bruckhaus	bruckhau	objective	object
brussels	brussel	objectives	objective
buddhist	buddha	observed	observe
buffs	buff	observer	observe
building	build	occasions	occasion
buildings	build	offering	offer
builds	build	offers	offer
built	build	official	office
bungalows	bungalow	oldest	old
burmese	burma	opening	open
business	busy	opens	open
businesses	businesse	operation	operate
busters	buster	operational	operate
butterflies	butterfly	operations	operate
bypass	bypas	operator	operate
bytes	byte	operators	operate
calling	call	opportunites	opportunite
calls	call	opportunities	opportune
campuses	campuse	opportunity	opportune
cams	cam	optimised	optimis
cards	card	options	option
cars	car	organic	organ
casuals	casual	organisations	organisation
categories	category	organization	organ
categorized	category	organize	organ
caters	cater	organized	organ
caught	catch	organizes	organ
cautious	caution	orgnizations	orgnization
cbs	cb	oriental	orient
cds	cd	original	origin
celebrations	celebrate	origins	origin
central	center	overseas	oversea
centre	center	pacific	peace
centres	center	packages	package

centuries	century	packed	pack
ceramics	ceramic	pages	page
ceremonies	ceremony	paintings	paint
certification	certificate	palaces	palace
certified	certifi	parents	parent
challenges	challenge	parks	park
chances	chance	parliaments	parliament
changed	change	partially	part
charters	charter	participation	participate
charts	chart	parties	party
checks	check	partnership	partner
chemicals	chemistry	parts	part
children	child	passports	passport
chinese	china	past	pass
choice	choose	pastes	paste
christian	christ	paths	path
christmas	christma	patronage	patron
cities	city	pcs	pc
claims	claim	penpals	penpal
classified	class	perks	perk
classifieds	classified	permissible	permit
classify	class	personal	person
clearance	clear	personalised	personalis
clinical	clinic	perspectives	perspective
clothing	cloth	petrochemicals	petroleum
clubs	club	pharmaceutical	pharmacy
coastal	coast	photo	photograph
collaboration	collaborate	photographed	photograph
collection	collect	photographers	photograph
colleges	college	photographs	photograph
coloured	color	photography	photograph
columbus	columbu	photos	photograph
comments	comment	phrases	phrase
commercial	commerce	picked	pick
commitment	commission	pictures	picture
committed	committ	piercing	pierce
communication	communicate	piercings	piercing

communications	communicate	pineapples	pineapple
community	commune	pins	pin
companies	company	places	place
companions	companion	planners	planner
competition	compete	plans	plan
compiled	compile	plants	plant
complementary	complement	plastic	plaster
complex	complicate	plastics	plaster
composed	compose	players	player
comprehensive	comprehend	pleased	please
computer	compute	policies	policy
computers	compute	political	policy
concept	conceive	politics	policy
concepts	conceive	pollution	pollute
concerning	concern	polymer	polymerize
concerns	concern	popular	people
conditions	condition	population	people
condominiums	condominium	portal	port
conducted	conduct	portals	port
conference	confer	portraits	portrait
conferences	confer	postage	post
confirmation	confirm	posted	post
congratulations	congratulate	postings	posting
connections	connect	poverty	poor
connectivities	connectivity	powered	power
conservation	conserve	powerful	power
consists	consist	practical	practice
consolidation	consolidate	practices	practice
constitution	constitute	prathes	prathe
constitutional	constitute	preferred	preferr
consular	consul	prepaid	prepay
consulate	consul	preparation	prepare
consultation	consult	preparations	prepare
consulting	consult	presents	present
contacts	contact	pressure	press
contains	contain	prevention	prevent
contemporary	contemporaneous	preventive	prevent

content	contain	prices	price
contents	contain	printer	print
continent	continence	privacy	private
continued	continue	privately	private
continues	continue	problems	problem
continuous	continue	produced	produc
conversion	convert	producer	produce
cookbooks	cookbook	productivity	produce
cooking	cook	products	produce
cooperative	cooperate	professional	profess
corporate	corporation	professionals	profess
corrections	correct	profiles	profile
costs	cost	programme	program
councils	council	programmes	program
countries	country	programs	program
courses	course	projects	project
courts	court	promotion	promote
coverage	cover	promotions	promote
covering	cover	proposed	propose
crafted	craft	prospective	prospect
crafts	craft	prosthodontics	prosthodontic
crags	crag	prostitution	prostitute
created	create	protection	protect
creation	create	proverbs	proverb
creative	create	provided	provide
credits	credit	providers	provider
crewed	crew	provides	provide
criteria	criterion	providings	providing
criterias	criteria	provinces	province
critics	critic	provisions	provide
crossroads	crossroad	public	publish
cruises	cruise	publication	publish
cuisines	cuisine	publications	publish
cultural	culture	publicity	publish
cultures	culture	published	publish
curators	cure	purity	pure
currencies	current	purposes	purpose

currency	current	quarters	quarter
currently	current	quations	quation
customer	custom	questions	question
customers	custom	quirkies	quirky
customized	customiz	quotes	quote
customs	custom	racketeers	racket
cuts	cut	raids	raid
cyclists	cycle	railways	railway
czech	czechoslovak	raised	raise
danish	dane	rapidly	rapid
dass	das	rated	rate
databases	database	rates	rate
davies	davy	reached	reach
daytrips	daytrip	reading	read
dealers	deal	reality	real
dealings	dealing	really	real
deals	deal	reasonable	reason
decades	decade	reasons	reason
declaration	declare	rebalances	rebalance
dedicated	dedicate	recipes	recipe
defense	defend	recommendation	recommend
delayed	delay	recommendations	recommend
delegates	delegate	recommended	recommend
delicious	delicate	recovery	recover
delivery	deliver	recruiters	recruiter
democratic	democracy	recruitment	recruit
demographics	demographic	reference	refer
demonstrated	demonstrate	refugees	refuge
dentist	dental	refunds	refund
dentistry	dental	regional	region
depicts	depict	regions	region
depth	deep	registration	register
derivatives	derive	reign	reich
described	describe	related	relate
description	describe	relations	relate
descriptions	describe	relationship	relate
designed	design	relationships	relate

destination	destine	releases	release
destinations	destine	relocation	relocate
detailed	detail	remained	remain
details	detail	remains	remain
developed	develop	renamed	renam
developer	develop	renewal	renew
development	develop	rent	rend
developments	develop	rental	rent
dictionaries	diction	rentals	rent
dictionary	diction	reports	report
different	differ	represents	represent
dips	dip	reproduction	reproduce
directly	direct	required	require
director	direct	requirements	require
directorate	direct	requires	require
directories	direct	researchers	researcher
directory	direct	reservations	reservation
disabled	disable	reserved	reserve
discerning	discern	residence	reside
disclaimer	disclaim	residential	reside
discos	disco	residents	reside
discounted	discount	resolution	resolve
discounts	discount	resorts	resort
discovery	discover	resources	resource
discreetly	discreet	restaurants	restaurant
discussion	discuss	results	result
disputes	dispute	retirement	retire
distilled	distil	retreats	retreat
distribution	distribute	returned	return
districts	district	reviewed	review
diversity	diverse	reviews	review
dives	dive	riders	ride
division	divide	rights	right
documents	document	rings	ring
downloads	download	ringtones	ringtone
dramatic	drama	rised	rise
dreaded	dread	risks	risk

drugs	drug	roads	road
durability	durable	rolls	roll
duties	duty	romania	roumania
easily	ease	rooms	room
eastern	east	rose	rise
easy	ease	roughly	rough
economic	economical	ruins	ruin
economics	economic	ruler	rule
economy	economical	ruling	rule
ecotours	ecotour	running	run
edens	eden	safe	save
edible	eat	safety	save
edited	edit	sale	sell
editor	edit	sales	sell
education	educate	samarts	samart
educational	educate	sandy	sand
educators	educate	sars	sar
elections	elect	saved	sav
electricity	electric	saving	save
electronic	electron	savings	save
electronics	electron	schedules	schedule
elephants	elephant	schools	school
embassies	embassy	searches	searche
embraces	embrace	seat	sit
emotional	emotion	sectors	sector
emphasize	emphasis	securities	secure
employee	employ	security	secure
employers	employ	seekers	seeker
employment	employ	seeks	seek
enabled	enable	selected	select
encyclopedia	cyclopaedia	selection	select
endangered	endanger	selections	select
engineer	engine	sells	sell
engines	engine	senators	senate
english	england	seniors	senior
enlargement	enlarge	sensation	sense
enroll	enrol	sensational	sense

enshrined	enshrine	sensory	sense
entered	enter	sensual	sense
entertainment	entertain	served	serve
enthusiast	enthuse	servers	server
entirely	entire	service	serve
entrance	enter	services	serve
entry	enter	setting	set
environment	environ	sexual	sex
equipment	equip	sexuality	sex
equipments	equip	shares	share
ers	er	shines	shine
escorted	escort	shops	shop
especially	especial	shorebirds	shorebird
essays	essay	shores	shore
essential	essence	shows	show
established	establish	siamese	siam
esthetics	aesthete	significant	sign
estimated	estimat	simpsons	simpson
ethnic	ethnology	singers	sing
evaluation	evaluate	sites	site
events	event	sits	sit
exaggeration	exaggerate	situation	situated
excellence	excel	sleeper	sleep
excellent	excel	slides	slide
exchange	change	slightly	slight
exciting	excite	smashed	smash
exclusive	exclude	smiles	smile
exclusively	exclude	sms	sm
excursions	excursion	snacks	snack
executive	execute	social	society
exercises	exercise	soldiers	soldier
exhibition	exhibit	solutions	solve
exhibitions	exhibit	songs	sing
exhibits	exhibit	sources	source
expedition	expedite	southern	south
experienced	experience	souvenirs	souvenir
experiences	experience	spas	spa

experiments	experiment	specialises	specialise
experts	expert	specialist	special
explicitly	explicit	specialists	special
exploration	explore	specialize	special
exporter	export	specials	special
exports	export	spectrum	specter
expression	express	speculative	speculate
faces	face	spent	spend
facets	facet	sponsored	sponsor
facial	face	sponsors	sponsor
facilities	facile	sports	sport
facility	facile	spots	spot
facts	fact	srs	sr
fails	fail	stainless	stain
families	family	stamps	stamp
fans	fan	standardization	standard
fares	fare	standards	standard
farmers	farm	staring	stare
farming	farm	stars	star
fatty	fat	started	start
favorite	favor	statement	state
favorites	favor	states	state
featured	feature	stations	station
features	feature	statistical	statistics
federal	federate	steps	step
fell	fall	sterilization	sterile
festivals	festival	stocks	stock
ffs	ff	stores	store
fifth	five	stories	storey
figures	figure	story	storey
fill	full	strategy	stratagem
filling	full	stretched	stretch
filmtrips	filmtrip	student	study
financial	finance	students	study
finder	find	studies	study
finding	find	stylish	style
findings	find	subjects	subject

finds	find	success	succeed
fingerprints	fingerprint	successful	succeed
finished	finish	successfully	succeed
firmness	firm	suggestions	suggest
firms	firm	suites	suit
flags	flag	sunday	sun
flight	fly	superbrands	superbrand
flights	fly	supervision	supervise
floating	float	supliers	suplier
floral	flora	supported	support
florists	florist	supporters	supporter
flowers	flower	surgery	surgeon
following	follow	survivor	survive
food	feed	systems	system
foods	feed	tailored	tailor
forecasts	forecast	tales	tale
forests	forest	targets	target
forgot	forget	teacher	teach
formal	form	teachers	teach
former	form	teamkites	teamkite
formerly	former	technical	technics
forums	forum	technologies	technics
forwarded	forward	technology	technics
forwarder	forward	ted	t
found	find	teenager	teenage
foundation	found	temperature	temper
founded	found	temples	temple
fractionated	fractionat	tens	ten
framed	frame	tenth	ten
frames	frame	terms	term
freedom	free	terror	terrible
freshfields	freshfield	texas	texa
friendly	friend	textile	texture
friends	friend	thai	thailand
friendship	friend	thailandtips	thailandtip
friendships	friend	thais	thailand
funded	fund	thaksinomics	thaksinomic

furniture	furnish	theatre	theater
gained	gain	things	thing
galleries	gallery	thoughtfully	think
gardens	garden	thoughts	thought
garnets	garnet	tickets	ticket
gates	gate	times	time
gathers	gather	timetables	timetable
gems	gem	tips	tip
gemstones	gemstone	tis	ti
generation	generate	titles	title
geographic	geography	tools	tool
geographical	geography	topics	topic
gibbons	gibbon	totally	total
gift	give	tourism	tour
gifts	give	tourist	tour
girls	girl	tourists	tour
given	give	tournaments	tournament
gives	give	tours	tour
global	globe	towns	town
globus	globu	toys	toy
golden	gold	tps	tp
golds	gold	traditional	tradition
golfcourses	golfcourse	traditions	tradition
golfer	golf	training	train
golftours	golftour	trains	train
goodness	good	transcripts	transcribe
government	govern	transition	transit
governmental	govern	transportation	transport
governments	govern	traps	trap
graduates	grade	traveler	travel
graphic	graph	travelers	travel
graphics	graphic	traveller	travel
grassroots	grassroot	travellers	travel
grimes	grime	travelog	travel
groups	group	travelogue	travel
growing	grow	travelogues	travel
growth	grow	treasures	treasure

guaranteed	guarantee	treated	treat
guardian	guard	treatment	treat
gueshouses	gueshouse	treatments	treat
guesthouses	guesthouse	treks	trek
guidebooks	guidebook	trends	trend
guided	guide	trial	try
guides	guide	tribes	tribe
habitats	habit	trips	trip
handicrafts	handicraft	tropical	tropic
hanns	hann	truly	TRUE
happiness	happy	tutorial	tutor
harbour	harbor	types	type
harriers	hare	typical	type
headlines	headline	typically	type
health	heal	underlined	underline
healthy	heal	understanding	understand
held	hold	understood	understand
helpful	help	unified	unifi
helping	help	union	unite
heritage	heritable	unit	unite
hermitage	hermit	united	unite
hidden	hide	universal	universe
highlands	highland	universities	university
highlights	highlight	unmatched	unmatch
highly	high	updated	update
hilltribes	hilltribe	updates	update
historical	history	useless	usele
hits	hit	users	user
holder	hold	vacancies	vacate
holidays	holiday	vacation	vacate
homes	home	vacations	vacate
honduras	hondura	valuable	value
honeymoon	honey	valuables	value
honeymoons	honey	variants	vary
horizontal	horizon	varied	vary
hospitality	host	variety	vary
hosts	host	various	vary

hot	heat	vas	va
hotels	hotel	vegas	vega
hots	hot	vehicles	vehicle
houses	house	veterans	veteran
housing	house	viewed	view
humanitarian	human	views	view
humidity	humid	villas	villa
hundreds	hundred	violator	violate
ideas	idea	visas	visa
identification	identical	visibly	vision
illustrated	illustrate	visited	visit
immediately	immediate	visiting	visit
immigration	immigrant	visitor	visit
implants	implant	visitors	visit
implemented	implement	visual	vision
important	import	volumes	volume
imports	import	volunteer	voluntary
impressions	impress	voted	vote
incentives	incentive	voyages	voyage
included	include	walks	walk
includes	include	warning	warn
inclusive	include	warnings	warn
inconvenience	inconvenient	waterways	waterway
increased	increase	wats	wat
independence	independent	wayfarers	wayfarer
indicators	indicate	ways	way
individuals	individual	webs	web
industrial	industry	websites	website
information	inform	wedding	wed
ingredients	ingredient	weekly	week
innovation	innovate	weeks	week
insects	insect	welcomes	welcome
insider	inside	western	west
insights	insight	whois	whoi
instant	instance	wholesalers	wholesaler
institution	institute	wickets	wicket
institutions	institute	wilfried	wilfri

insurance	assure	winner	win
integral	integer	winning	win
intellectual	intellect	wired	wire
intended	intend	wires	wire
interested	interest	witnessed	witness
interesting	interest	women	woman
interests	interest	won	win
internationale	international	wonderful	wonder
intimacy	intimate	words	word
introduction	introduce	workers	work
investigations	investigate	works	work
investment	invest	workshops	workshop
investments	invest	writers	write
investor	invest	writings	write
investors	invest	written	write
involved	involve	xmas	xma
islands	island	yeaars	yeaar
issued	issue	zoological	zoology

BIOGRAPHY

NAME	Mr. Kovit Karawatreedech
DATE OF BIRTH	01 June 1972
PLACE OF BIRTH	Chulalongkorn Bangkok, Thailand
INSTITUTIONS ATTENDED	Mahidol University, 1994-1995: Bachelor of Science (Computer Science) Mahidol University, 1999-2005: Master of Science (Computer Science)
POSITION&OFFICE	1996-Present, Reuters Software Thailand 30th Floor U Chu Liang Building 968 Rama IV Road Silom, Bangrak Bangkok 10500 Thailand Position: Senior Software Engineer