A Generic Chinese PAT Tree Data Structure for Chinese Documents Clustering

KWOK Chi Leong

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Philosophy
in
Department of Computer Science & Engineering

Supervised by:

Prof. Michael R. LYU

A Generic Chinese PAT Tree Data Structure for Chinese Documents Clustering

submitted by

KWOK Chi Leong

for the degree of Master of Philosophy
at The Chinese University of Hong Kong
in March 2002.

Abstract

Chinese information processing technique is one of the important aspects in the computer science research because Chinese is one of the popular languages besides English, especially in the Asia Pacific region. The number of electronic Chinese documents is growing rapidly due to the fast expansion of the Internet. There is an urgent demand on techniques for Chinese information retrieval and archiving. In our research, we propose a generic Chinese PAT tree in handling Chinese documents. By applying our generic Chinese PAT tree, we develop a system framework for Chinese documents clustering.

PAT tree is a compact structure that can represent documents efficiently. The PAT tree structure has also been applied to Chinese documents. The Chinese PAT tree is derived to provide basic functionality in handling Chinese documents. We propose a generic Chinese PAT tree that features the new concept of Essential Node to improve the capability of the structure. The generic Chinese PAT tree can capture information for document phrase for various purposes like segmentation, word extraction, clustering, etc. With the concept of Essential Node, every Chinese phrase in the document must be associated with a unique tree node. This makes our generic Chinese PAT tree capable of handling Chinese phrases of correct syntax properly. We evaluate the performance of various PAT trees. The result shows that our generic PAT tree approach consumes not much higher computation time than the other PAT tree structures. Theoretically, our generic Chinese PAT tree has the same storage and runtime complexity as the other PAT trees. It is a $O(n)$ structure that can perform searching in $O(\log n)$ and traversing in $O(n)$ on average.

During the study of PAT tree, we discover a critical issue of large computation overhead when using the "dynamic memory" implementation for the PAT tree. We consequently propose a node production factory implementation approach that can overcome this problem. Our evaluation shows that the PAT tree structure using the node production factory implementation approach is practical and feasible.

With the help of the generic Chinese PAT tree, we define a system framework for Chinese documents clustering. The framework benefits from the generic Chinese PAT tree in handling Chinese documents. We identify base clusters from the documents collections. The system analyzes and filters the base clusters to combine the result into final clusters of documents. From our documents clustering experiment, our clustering framework is capable of discovering large clusters and clusters with outstanding topics from the documents collection. We further evaluate our clustering framework with documents collection under different situations. The evaluation shows that using part of documents as the abstract can produce suitable clustering result in faster response time. The result provides positive argument to employ our proposed documents clustering framework into the online Web environment in the future.

A Generic Chinese PAT Tree Data Structure for Chinese Documents Clustering

submitted by

KWOK Chi Leong

for the degree of Master of Philosophy
at The Chinese University of Hong Kong
in March 2002.

論文摘要

中文資訊處理方法是計算機科學研究中的一個重要課題，因為尤其是在亞太地區，中文是英文以外的一種流行語言。由於互聯網急速擴展，中文電子文件的數目迅速地增長。這對中文資訊搜尋及儲存技術有著迫切的需求。我們在研究中提出一個一般性中文 PAT 樹狀結構來處理中文電子文件。應用這結構，我們發展了一個建立中文文件群集的系統架構。

PAT 樹狀結構能簡潔有效地描述文件，它亦有被應用到中文文件上。這衍生出來的中文 PAT 樹狀結構能提供處理中文文件的基本功能。我們提出的一般性中文 PAT 樹狀結構則包含了「重要樹結」這個新概念來增強結構的表達能力。一般性中文 PAT 樹狀結構能捕捉文章詞句的相關資訊作各種用途如：文句分割、文字提取、群集分析等。藉著重要樹結這概念，每個詞句都會聯繫到獨一無二的樹結上，令這結構能夠適當地處理中文文法。我們評估了不同 PAT 樹狀結構的效能，結果顯示一般性中文 PAT 樹狀結構並不會比其他 PAT 樹狀結構消耗太多運算時間。理論上，它們全都是 O(*n*) 複雜度的結

構，而且能夠在 O(log$n$) 時間內作出搜索以及在 O(n) 時間內作出結構全覽。

我們在研究當中發現，使用動態記憶配置方法去建立 PAT 樹狀結構會嚴重影響其執行速度。我們因此提出了樹結製造廠這配置方法。在效能評估中顯示，這方法能實用和合理地克服這問題。

我們利用一般性中文 PAT 樹狀結構來協助，定義這個建立中文文件群集的系統架構。它能以這結構去處理中文文件。從文件集之中辨別出基礎群集後，該系統會進行分析和篩選並綜合出最終的群集結果。從實驗結果顯示，該系統能夠發掘出大群和有特色主題的群集。在進一步的評估中發現，我們若使用文件摘要部分作分析，便能夠加快計算過程，而且能夠產生更合適的群集結果。研究結果提供了確實論據顯示該系統能夠將來應用在互聯網線上環境。

Acknowledgment

I would like to take this opportunity to express my gratitude to my supervisor, Prof. Michael R. Lyu, for his generous guidance and patience to me during the research time.

I would like to thank Prof. Irwin King for giving ideas, support, and encouragement to me. The inspiring advice from Prof. Lyu and Prof. King are extremely essential and valuable in my research paper, which is published in The International Symposium on Information Systems and Engineering 2001 (ISE 2001), as well as my thesis.

I am also grateful for the time and valuable suggestions that Prof. Ada Fu have given in marking my term papers.

I would also like to show my gratitude to the Department of Computer Science and Engineering, CUHK, for the provision of the best equipment and pleasant office environment for high quality research.

I want to give my thanks to my fellow colleagues, C. Y. Chan, C. C. Cheung, Vincent Cheung, H. C. Ho, Norris Leong, Ivan Leung, Jacky Ma, Nicky Ng, Raymond Pang, Polly Wan, Cliff Sze, Keith Wong, Kevin Yuen, and W.S. Yuen. They have helped me in solving technical problems, enlightened me with new research ideas, and given me encouragement and supports. They have given me a joyful and wonderful time in my research.

Finally, my special thanks must go to my CUHK CSE programming team members, L. C. Lau and Starsky Wong, who have given me an unforgettable memory in my university life.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Chinese information processing technique is an important aspect in the computer science research because Chinese is one of the most popular languages. With the large population of Chinese people and the fast expansion of the Internet and the World Wide Web, the number of electronic Chinese documents is growing rapidly. Despite the format and nature of Chinese characters, usual techniques that can be applied to English information [41][43] cannot be directly adopted to handle information in Chinese [13][32][37]. There is an urgent demand on the techniques for Chinese information archiving and retrieval.

In the recent research on the Chinese information processing, many of the research efforts focused on Chinese keyword extraction [10][14][16], sentence segmentation [8][22][25], and document searching [11][12]. They combine the understanding of Chinese information with fundamental techniques for text retrieval [4][5][19], pattern matching [1][49], and word understanding [18][47].

From the topic of Chinese information processing, we focus on the documents clustering problem in Chinese documents collections. Although the clustering problem is a common topic in various domains, there is not much research in the topic of documents clustering in Chinese documents collections. Our goal then is to discover clusters from the documents collection with common topics. Those topics should be the most popular topics among the set of documents collection. Sometimes these topics are outstanding phrases that represent a group of documents.

We propose a generic Chinese PAT tree data structure which benefits from the design of PAT tree and is capable of handling Chinese information for the clustering need. We further formulate a documents clustering framework that makes use of the generic Chinese PAT tree data structure to conduct the Chinese documents clustering. This clustering framework runs in linear time and is suitable for real-time execution. Therefore, our document clustering solution is also appropriate for clustering Chinese documents on the Web.

From the analysis of our generic Chinese PAT tree and experiments on our clustering framework, the generic Chinese PAT tree is efficient in storing phrases of documents correctly. The structure is able to maintain statistical information from the clustering framework. Results from our clustering experiments demonstrate the applicability of clusters that our framework produces.

1.1 Contributions

Our research work makes the following contributions:

1. We have proposed a generic Chinese PAT tree data structure. The structure has linear storage complexity and has features of PAT tree. It is an index structure that is capable of handling Chinese information. Our generic design of the tree makes the structure itself consistence and evolvable.

2. We have proposed a definition of Essential Node (EN) to indicate the phrasal information in the generic Chinese PAT tree. Each Chinese phrase inside the generic Chinese PAT tree is represented by one tree node, which is called Essential Node. Essential Nodes are a subset of tree nodes that are effective for storing Chinese phrases.

3. We have proposed a node production factory approach to address the

overhead issues in the memory management of PAT tree implementation. We evaluate implementations of PAT tree that use static and dynamic memory management approaches. The dynamic approach is common but suffers from very large non-linear overhead in PAT tree. Our node production factory approach is a solution to reduce such overhead and make it linear to the documents processed.

4. We have proposed a Chinese documents clustering framework that makes use of the generic Chinese PAT tree data structure. The clustering framework can discover interesting groups of documents for a particular set of documents collection.

5. We have implemented a system to demonstrate our work. The system can collect news articles in advance from the Web as the documents collection. It produces resulting clusters from the supplied news articles, which highlight the interesting news topic among the given set of news.

1.2 Thesis Overview

We explain our research work in details in the following chapters.

In Chapter 2, we describe several work that is related to the clustering and Chinese information processing. We review several clustering techniques and introduce the suffix tree clustering, which is used as the reference model of our clustering framework. In the area of Chinese information processing, we describe typical problems and solutions in recent research.

In Chapter 3, we detail the structure of PAT tree and the requirement of the Chinese PAT tree. We propose our generic Chinese PAT tree data structure with detailed explanations and illustrations. We compare an existing design of PAT tree for Chinese, which is the embedded node design structure, with our generic structure design, and discuss the pros and cons of the generic Chinese PAT tree.

In Chapter 4, we evaluate the performance of PAT trees and discuss their implementation issues. We address the overhead problems due to the implementation issue of the PAT tree. We propose a node production factory approach to overcome this overhead problem.

In Chapter 5, we describe our Chinese documents clustering framework. We discuss the work in each of the clustering process. Several approaches for the base clusters detection and base clusters filtering process with their pros and cons are explained in more details in this chapter.

In Chapter 6, we evaluate our documents clustering framework. We apply our clustering system based on the proposed framework on a set of online news articles from the Web. We illustrate the effectiveness of our clustering result and demonstrate the impact of the result from document sets on different situations.

We conclude our work in Chapter 7.

Chapter 2

Background Information

In this chapter, we discuss some clustering techniques and the research related to Chinese information processing. In the clustering part, we introduce some generic clustering techniques and their problems when applied to documents clustering. We then introduce a specific clustering method, suffix-tree clustering (STC), that performs better in document clustering problems when compared with the generic clustering techniques. In the Chinese information processing part, we give an overview on the related research for the sentence segmentation problem and the keyword extraction problem. These problems provide us a better understanding of difficulties on Chinese information processing, and give us an idea on how PAT tree may be applied to benefit from these problems.

2.1 Documents Clustering

2.1.1 Review of Clustering Techniques

The documents clustering problem is a study of algorithms that analyze a collection of documents to identify clusters among them. There are quite a lot of literatures applying different clustering techniques and algorithms to perform documents clustering [7][26][28][50]. Typical clustering methods include Agglomerative Hierarchical Clustering algorithm [48], K-means algorithm [42], Single-Pass method [27][40], and Buckshot and Fractionation [21], etc. Each of these clustering algorithms is well known and each of them has their strengths as well as their limitations.

Agglomerative Hierarchical Clustering algorithms are usually slow when the size of the document collection becomes large. This method takes $O(n^2)$ time with the single-link or group-average methods while it takes $O(n^3)$ time with the complete-link methods.

K-means algorithm is a linear time clustering algorithm with $O(nkT)$ time complexity where k is the number of desired clusters and T is the number of iterations. This algorithm can produce overlapping clusters but it is only effective when the clusters are in spherical sharp with respect to the similarity measurement. Therefore, K-means algorithm may not perform well in realistic situation on document clustering.

The Single-Pass method is also a linear clustering algorithm similar to the K-mean algorithms. Besides the disadvantages mentioned in the K-mean algorithms, the Single-Pass method also suffers from the disadvantages of order dependency and tendency in producing large clusters.

Buckshot is a K-means algorithm where the initial cluster centroids are created by applying the Agglomerative Hierarchical Clustering to a sample of the documents of the collection. It is a fast and linear time clustering algorithm but not incremental. Besides, it is not suitable for identifying small clusters.

Fractionation is an approximation to Agglomerative Hierarchical Clustering algorithms but the search for the two closest clusters is performed locally and in a bound region. It suffers the same disadvantages from the Agglomerative Hierarchical Clustering algorithms.

Besides the previously mentioned clustering algorithms, Zamir [56] suggests another clustering algorithm that is suitable for documents clustering. That approach is known as suffix-tree clustering (STC), which is a special clustering technique for document clustering problem. According to Zamir, STC is especially suitable on clustering the Web documents due to its incrementally property.

2.1.2 Suffix Tree Clustering

Suffix Tree Clustering (STC) is a novel, incremental, O(*n*) time algorithm proposed by Oren Zamir and Oran Etzioni in 1998 [56]. It relies on a suffix tree to identify the sets of documents that share common property and applies this information to create clusters.

Unlike generic clustering approaches, the STC approach makes use of the nature of documents for clustering. It does not treat a document as a set of words. Instead, STC treats a document as string and makes use of proximity information between words. This is one of the main reasons why STC is more suitable for documents clustering compared with other generic clustering approaches.

The research on Web technology and search engine shows demand on the documents clustering. Traditionally, techniques of documents clustering can pre-cluster the entire corpus in order to improve the search engine in terms of performance [41]. Besides, techniques of documents clustering can also be used as a post-retrieval documents browsing technique [2][20][21][30]. Since STC is an incremental clustering algorithm, it is suitable for clustering Web documents.

According to Zamir, the clustering result using STC approach out performs the result using other generic clustering techniques. He conducted a series of comparison in terms of average precision and time. In his result scale, the other clustering techniques obtain their average precision varying from 0.21 to 0.36, and STC has better average precision at about 0.39. On his run time execution measurement, STC is comparable to any of the linear time clustering techniques. In his experiment environment, the execution time of STC for a collection of up to one thousand documents is around 7 to 15 seconds, which is in an acceptable range for real-time clustering applications.

By observing the strengths of STC in documents clustering, we base our proposed Chinese documents clustering framework on this method. We apply our proposed generic Chinese PAT tree data structure to handle the Chinese information and apply several Chinese information processing techniques inside the clustering framework to produce a feasible solution to the Chinese documents clustering. We describe the details of our Chinese documents clustering framework in Chapter 5.

2.2 Chinese Information Processing

Early techniques on Chinese information processing are adopted from the techniques in English. These techniques are limited to applications on Chinese characters. These approaches do not make much sense with the nature of the Chinese language. Nie [37] and Chen [9] also suggest that the Chinese information processing techniques should focus on Chinese words instead of Chinese characters. However, the contextual information of Chinese sentences provides no clue about Chinese words. This problem reveals the difficulties of the research of Chinese information processing.

2.2.1 Sentence Segmentation

Chinese sentence segmentation is one of the fundamental problems in Chinese information processing [31][33]. Many researchers have studied this problem [8][25][51][54] and their solutions are mainly in two different approaches: the dictionary approach and the statistical approach [44][45][53].

Early work on the sentence segmentation problem focuses on the dictionary approach. It makes use of the general Chinese words knowledge and lexical concepts to identify Chinese words. Since the dictionary approach is understandable and human-oriented, we could easily verify the correctness of results from this approach. However, vocabularies are evolving from time to time, and we are unable to guarantee the existence of all possible words in one dictionary. The dictionary in this approach cannot be complete no matter how large it can be. Therefore, the dictionary approach performs poorly in the environment with many newly evolved vocabularies and names, which are not contained in the dictionary.

The statistical approach is a more sophisticated solution and it is more and more popular recently. Nie [37] attempts to use the statistical approach to complement the dictionary approach and comes up with a hybrid approach solution. He suggests adopting the dictionary approach that would assign each existing word with a default probability value. He tries to discover potential vocabularies and new words from the statistic results in training data sets. Every potential vocabulary and new word associated with a probability obtained from the knowledge during the data training. He uses this probability information to obtain the best segmentation scheme on the Chinese texts, which gives the highest value of the product to their probabilities. The hybrid approach can provide additional statistical information for the problem when the dictionary knowledge is lacking. His study emphasizes more on using the statistical information with dictionary as backup on the sentence segmentation problem.

Chen [9] suggests a pure statistical method on sentence segmentation without using a dictionary. He builds the index list of words with one character long with their occurrence frequencies. A list of potential words with n characters can be generated from the list of potential words that have $(n-1)$ characters. He applies the Berkeley adhoc formula developed by Cooper [19] on his solution to obtain the desired sentence segmentation scheme. Later on, Dai [22] improves over Chen's approach [9] on the method of identifying the two-character words. He shows a better segmentation result when there is ambiguity during the segmentation. His solution tends to identify the meaningful words by considering the text context, rather than according to simple words from the dictionary.

Summarizing these approaches, there are mainly three different ways to solving the problem. These approaches are namely the dictionary approach, the statistical approach, and the hybrid approach. The comparison of them can be found in Table 2.1.

Dictionary Approach	**Hybrid Approach**	**Statistical Approach**
Vocabularies		
➢ Vocabularies are from the selected dictionary ➢ The number of vocabularies is limited by the size of the dictionary	➢ Vocabularies are from training data ➢ Has additional dictionary as a back-end support	➢ No predefined knowledge of vocabularies ➢ Vocabulary sets are learnt and generated from data text
Methods		
➢ Simple matching rules can be applied directly ➢ (e.g., maximum matching)	➢ Use best probability searching with probability value from the training data ➢ The dictionary help for pre-calculation or tie breaking	➢ Usually based on an artificial measurement functions ➢ The method may be non-human natural
Advantages		
➢ Human understandable ➢ Easy to implement	➢ Give reasonable accuracy from combining both approaches	➢ More accurate result can be obtained with an appropriate function model
Suitable Situation for Segmentation		
➢ Text with particular topics ➢ Has well-defined dictionary to refer to	➢ Text with particular topics ➢ But the topic has evolving vocabularies over time	➢ General text ➢ Text involves unusual or technical terms in no particular field
Disadvantages		
➢ There is no ideal dictionary. ➢ Result is poor when no words could be matched with the supplied dictionary	➢ Tuning the weighting from two approaches is quite complicated ➢ Difficult to balance the two approaches well	➢ The statistic function can be quite complicated in order to obtain a better accuracy

Table 2.1 Comparisons of different sentence segmentation approaches

2.2.2 Keyword Extraction

Keyword extraction is a critical problem in Chinese information processing

[16][17]. While the sentence segmentation is an effort to partition a single document into a bag of words with sequence, keyword extraction is an approach to identify the outstanding words from a single document or a group of documents within a scope. Keywords are concepts of abstract and summarization. They are main words to refer a text or document. Although keywords are commonly used in everyday life to represent complicated content such as document text, people usually need to produce and identify keywords manually. Automatic extraction of keywords is not common because the automatic extraction techniques are difficult and there is not much successful research in this area [10].

There are several techniques in handling the Chinese keyword extraction problems. One of the methods is to make use of the character-level information to construct a keyword extraction system [11]. The system replaces the word-level information and ignores the concept of words. Another method is to apply the lexicon analysis to perform the extraction process automatically [37][51]. This method involves sophisticated sentence segmentation techniques as well as some rigid lexicons and linguistic analysis for word-level information.

Because of the high complexity in the linguistic analysis and sophisticated sentence segmentation techniques, Chien [14] suggests a statistical method for the keyword extraction on the Chinese information. His method uses the PAT tree data structure to perform the indexing and applies the statistic functions to detect the resulting keywords. He avoids complicated linguistic measurement and employs the word-level information embedded in the PAT tree structure. His idea illustrates the capabilities of PAT tree data structure to help in solving some problems in Chinese information processing.

Chapter 3

The Generic Chinese PAT Tree

PAT tree [24] is a data structure that addresses the storage and representation issues of documents. Since the original design of PAT tree considers documents only in English, it cannot apply to Chinese documents directly. Chien proposed a variation of PAT tree in [14] for Chinese documents. That variation is in embedded node design, which combines the leaf nodes with the internal nodes, and facilitates the usage of occurrence frequency. His structure applies in different Chinese information processing applications including Chinese sentence segmentation and Chinese keyword extraction. However, due to the embedded node design of the structure, that Chinese variation has its own limitations and is unable to expend or enhance on its functionality. Therefore, by adopting the idea of occurrence frequency from that Chinese variation, we derive our Chinese PAT tree data structure in a non-embedded design with the same functionality. Our structure is a $O(n)$ storage complexity structure that can represent sub-string information. The searching of the sub-string information can be done in $O(\log n)$ time

In the design of our Chinese PAT tree, we extend from the basic functions of Chien's Chinese PAT tree and propose the generic Chinese PAT tree. The generic Chinese PAT tree not only includes all the features and functionality of the fundamental PAT tree structure, but also facilitates the ability in handling Chinese information. A new feature in this proposed structure is the capability to represent Chinese phrases in tree node with statistical information. This idea improves the usage of PAT tree in Chinese information application. The generic Chinese PAT tree will be used in our proposed Chinese document clustering technique.

In this chapter, we first detail the PAT tree data structure and the requirement to support Chinese, and present our basic variation of Chinese PAT tree. Based on our basic variation of Chinese PAT tree, we introduce the generic Chinese PAT tree. We keep the figures and descriptions in a consistence way in the PAT tree, the Chinese PAT tree as well as our generic Chinese PAT tree. Afterwards, we discuss the details and problems in the embedded node design in Chien's variation. A brief summary of differences between these structures is given in the last section of this chapter.

3.1 PAT Tree

PAT tree [24] is a data structure that is widely used in the area of information processing. It is a Patricia tree [36] that stores every semi-infinite strings (sistrings) [23] of a document into the leaf nodes of the tree.

3.1.1 Patricia Tree

Patricia Tree is a condensed trie that ensures all parent nodes must have exactly two child nodes. We can obtain a Patricia Tree from tries by removing nodes that contain only one single child. The links pointing to those removed nodes will redirect to the child of that removed node. Since all nodes with single child are removed (in other words, "merged with its parent"), Patricia Tree is very compact.

An example of trie and Patricia tree storing 2 (010_2), 3 (011_2) and 5 (101_2) are shown in Figure 3.1. The corresponding trie maintains the binary digital tree structure to represent the three numbers. Since the structure of the trie itself is sufficient to represent its content, node content is optional and may be omitted. By removing nodes with one child (the shaded nodes), we obtain the corresponding Patricia tree with equivalent content. Patricia tree requires index information in each node since the structure layout is not sufficient to provide all the tree content. However, Patricia tree requires significantly fewer nodes, which ensures a better construction time and search time.

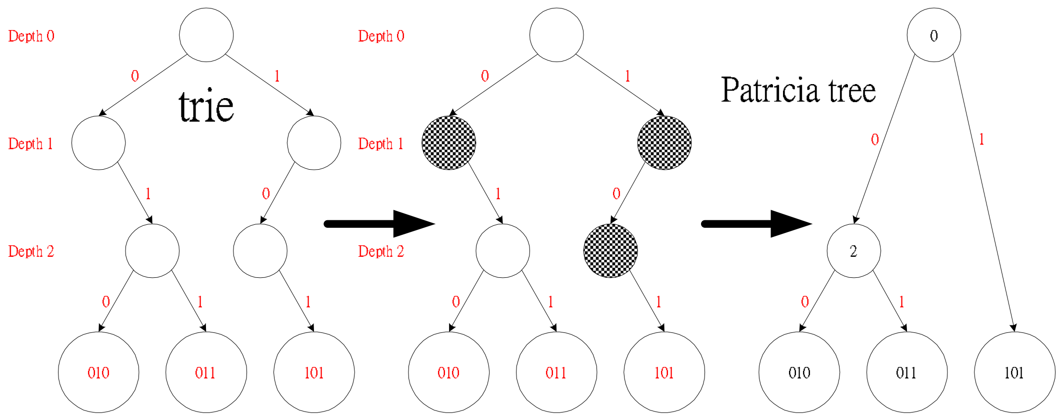


Figure 3.1 Illustration of trie and Patricia tree

3.1.2 Semi-Infinite String

Semi-Infinite String (Sistring) [23][34] is a subsequence of a string, taken from an arbitrary starting point to the every end of the right, padded at end with infinite number of null symbols. Conceptually, sistring, by its nature, would have infinite length. However, no matter how much capacities do we have, it is not possible to store a string of infinite length in computer memory. Therefore, in practice, we may not physically store sistring in the form of infinite length. We usually append the sistring with just enough number of null symbols, depending on applications and requirements, in actual implementation. In general, the length of a sistring should always be linear to the length of its original string.

Semi-Infinite String is useful in representing sub-strings information in a document efficiently. We can treat a document as a single string with a long length. When the document has *n* elements (characters), the number of all possible sub-strings would be $O(n^2)$. Since the length of each sub-string can be up to *n*, for a document with *n* characters, we require $O(n^3)$ memory to store all possible sub-strings. This $O(n^3)$ storage requirement is impractical in most cases because, not to mention a full document, even one sentence in English may already contain hundreds of characters. With the aid of semi-infinite strings, however, we can represent all sub-strings properly. Since a string with n elements (characters) will only include n sistrings, the storage requirement could be reduced to $O(n^2)$.

To show the effectiveness of sistrings representation, Table 3.1 and Table 3.2 give examples of sub-strings and equivalent sistrings representation over a simple document. Table 3.1 illustrates all the sub-strings of the simple document contains a single string '`computer`'. The example string has 8 characters. The total number of sub-strings for the string is 36 ($8+7+6+5+4+3+2+1=36$). Instead of explicitly storing all these 36 sub-strings individually, we store a list of all eight sistrings, each of them representing one or more than one sub-strings. Every possible sub-string is an arbitrary prefix of a sistring, and the former would be represented by the latter. The details are shown in Table 3.2.

i	**List of sub-strings of length *i***	**Number of sub-strings**
1	`c o m p u t e r`	8
2	`co om mp pu ut te er`	7
3	`com omp mpu put ute ter`	6
4	`comp ompu mput pute uter`	5
5	`Compu omput mpute puter`	4
6	`comput ompute mputer`	3
7	`compute omputer`	2
8	`computer`	1

Table 3.1 List of sub-strings of eight-character string: '`computer`'

Sistring	**Corresponding Sub-strings represented**
`computer`	`c co com comp compu comput compute computer`
`omputer0`	`o om omp ompu omput ompute omputer`
`mputer00`	`m mp mpu mput mpute mputer`
`puter000`	`p pu put pute puter`
`uter0000`	`u ut ute uter`
`ter00000`	`t te ter`
`er000000`	`e er`
`r0000000`	`r`

Table 3.2 Sistrings of eight-character string: '`computer`'

We notice the effect of the padded null symbol (`0`) in sistring. For the example in Table 3.2, the sistrings would be stored into equal length and so they are null-padded up to the one with maximum length. This is a tactical concern to ensure all sistrings are fully aligned so that they are directly comparable to each other. Although we can still pad more null symbols for the same effect, the extra null symbols have no added value but occupying more memory. Therefore, by assessing this example, we can verify that the storage requirement here is $O(n^2)$.

All possible sub-strings can be obtained by performing a prefix searching on the full set of sistrings. For example, if we are looking for the target string '`put`' in the document with a single string '`computer`', we can perform the prefix searching on sistrings as in Table 3.2. We illustrate the prefix searching process in Figure 3.2. The process searches through the list of sistrings one by one. As long as we discover mismatch of character in one sistring, we skip to the next sistring to continue our searching. The target string will match on a prefix of sistrings if and only if the target string is a sub-string of that document. The run time complexity of this prefix searching, in the worse case, would be $O(mn)$ where m is the length of the target string and n is the number of sistrings. However, when the sistring does not match with the target string, we will likely find a mismatch before reaching the last character of the target string and skip to compare the next sistring. Therefore, the actual run time would be much less then $O(mn)$ in general.

Step	Target String	Sistring to be compared with	Character Match or not?	Sub-string Found?
1	`put`	`computer`	No	
2	`put`	`omputer0`	No	
3	`put`	`mputer00`	No	
4-a	`put`	`puter000`	Yes!	
4-b	`put`	`puter000`	Yes!	
4-c	`put`	`puter000`	Yes!	Yes!

Figure 3.2 Prefix matching of '`computer`' with target '`put`'

3.1.3 Structure of Tree Nodes

The structure of PAT tree is similar to that of a usual Patricia tree. The major difference between them is the type of content to be stored. Since PAT tree is the Patricia tree that stores every sistring of a document, the leaf node in PAT tree is designed to store sistrings rather than general items like numbers.

There are two types of node in PAT tree data structure: the internal nodes and the external nodes. These two types of nodes are storing different things and have different node structures. External nodes are actually leaf nodes of the tree that contains information about the sistring. Internal nodes are the non-leaf nodes that serve for the indexing purpose. As sistrings plays an important role in the PAT tree, we further discuss sistrings in this section.

3.1.3.1 The Internal Nodes

Index information of PAT tree is inside each internal node. Each internal node contains two types of basic elements: a check bit and tree pointers to its children. The internal node of PAT tree, from definition of Patricia tree [24], always links to exactly two child nodes. Figure 3.3 outlines the basic structure of an internal node of the PAT tree.

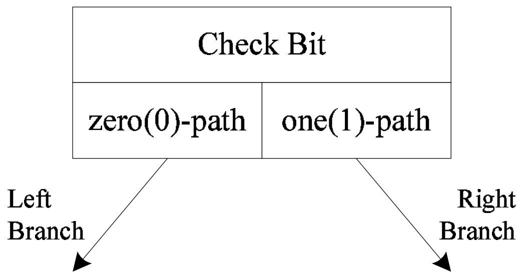


Figure 3.3 Structure of internal node of the PAT tree

Tree pointers are linked to their left child as well as their right child. Child node can be either an external node or an internal node. That means tree pointers are just ordinary tree pointer that can link to either an internal node or an external node. By convention of PAT tree, we sometimes call the left branch of the tree pointers zero-path while the right-branch of the tree pointers one-path. This convention can better reflect the meaning and usage of these tree pointers.

The check bit is a number denoting a bit position. Depending on the check bit value, we can classify the sistrings under an internal node into two partitions. By inspecting the bit value of the sistrings at the position indicated by the check bit, the sistrings are stored either in the partition of zero(0)-path or in the partition of one(1)-path. Besides, the check bit of the internal nodes has an order restriction. Check bits of the internal nodes must be in strictly increasing order along the tree path. In other words, if the child node of an internal node is also an internal node, the check bit of that child node must have a larger value than that of its parent node.

3.1.3.2 The External Nodes

As we go along the path of the PAT tree, each internal node separates the sistrings into two partitions until we reach an external node. External nodes are the leaf nodes of PAT tree so that they have has no further tree branches. While each internal node always partitions sistrings into two groups, each external node always uniquely represents one sistring.

Since sistring is not a simple element but the form of a string, external nodes do not store the sistring directly. Instead, external nodes contain a pointer, which refers to the location of the sistring. Therefore, the basic element in an external node is just the pointer to the represented sistring. Figure 3.4 shows the basic structure of the external node of PAT tree.

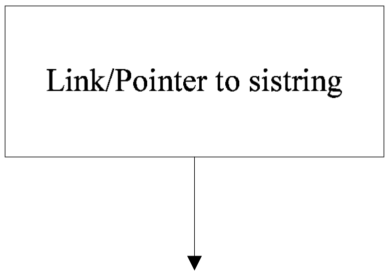


Figure 3.4 Structure of external node of the PAT tree

In some literatures, people suggest the external node should be combined with one of the internal node. This means that each internal node would have an additional pointer to sistring to simulate the same function as an external node. Under this structure design, the tree will not end at some leaves, but pointing back to an existing internal node. Since the check bit along the tree path of internal node must be in strictly increasing order, the check bit is the indicator to denote the nature of the node. When the check bit along the tree path agrees with the strictly increasing nature, it will be generally an internal node. When the check bit violates this rule, that corresponding node will not be a valid internal node in that context. It acts as an external node for that path. This is the embedded node design that merges external node into an existing node, so it can reduce the number of nodes by half. However, the storage complexity in this reduced design does not improve and the actual memory requirement does not change. On the other hand, this reduced design makes the PAT tree looks complicated. The structure becomes difficult to understand. The operations, like the tree construction process, are more difficult to handle. More seriously, it makes the PAT tree more difficult to evolve and to compare with its variations. Detailed discussions about the embedded node design of PAT tree are in Section 3.4.

3.1.3.3 Implicit Sistrings Representation

Storage complexity for sistrings of a string with n elements is $O(n^2)$. We have already discussed about this storage requirement in Section 3.1.2. However, the quadratic complexity becomes the bottleneck for PAT tree. Instead of storing the sistrings explicitly in $O(n^2)$ storage, we try to represent the sistrings implicitly with the document. We use a simple document with one string 'CUHK' as an example and illustrate the idea of implicit sistrings in Figure 3.5. When the sistrings of 'CUHK' is explicitly present, we require $O(n^2)$, with n equals 4, memory for the sistrings. However, under the implicit representation, we store the original document in the memory padded with virtually n more null symbols. Although the sistrings are now fully embedded inside the document, we can still have sistrings in equal length n. This approach effectively maintains the equivalent sistrings information, but requires only $O(n) + O(n)$ memory, which keeps a linear storage of $O(n)$. As a result, the pointers in the external nodes, in both approaches, are referring to indifferent kinds of information.

Instead of storing the sistrings explicitly, the external node will use the pointer pointing to the document at the same position as the beginning of the representing sistring. Therefore, we have a list of sistrings embedded in the PAT tree structure without occupying extra memory for physically storing all sistring, which is $O(n^3)$. In order to maintain the semi-infinite property, the original document can be null padded to double the original document size. As a result, although all sistrings are still embedded inside the document, they can still be of equal length up to n.

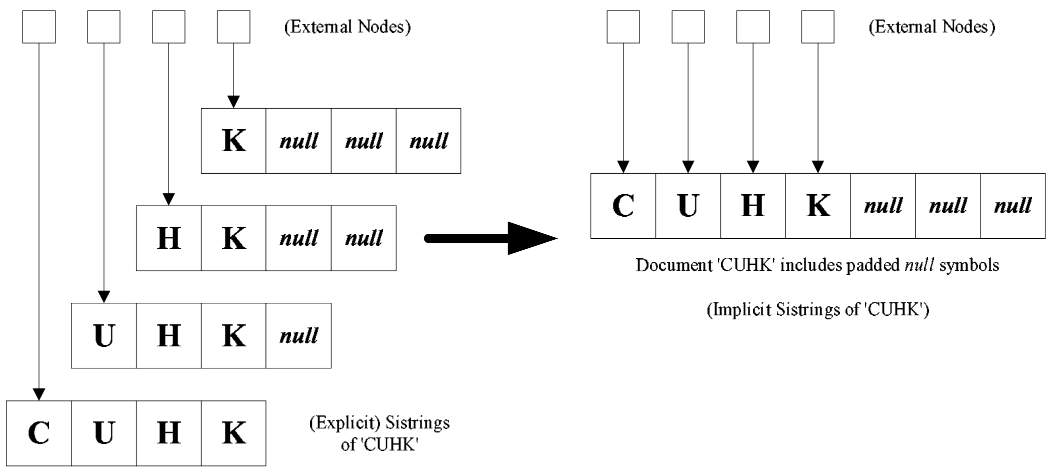

(External Nodes)
K null null null
H K null null
U H K null
C U H K
(Explicit) Sistrings of 'CUHK'
(External Nodes)
C U H K null null null
Document 'CUHK' includes padded null symbols
(Implicit Sistrings of 'CUHK')

Figure 3.5 Explicit vs. implicit sistrings representation

In conclusion, PAT tree has the tree nodes and its associated sistrings. Each kind of tree nodes must contain the basic information as mentioned above. Because of the design of the tree structure, PAT tree contains the following properties:

1. When the internal node includes a check bit with value *b*, the sistrings under the node branches are partitioned exactly into two groups. One group of sistrings, with value '0' at bit-*b* position, is stored under the left branch (zero-path). Another group of sistrings, with value '1' at bit-*b* position, is stored under the right branch (one-path).

2. We need not backtrack the branches when we are searching for a particular sistring. It is because the branching decision in each internal node is well-defined. We can ensure that, in each internal node, the target sistring may exist in one branch but it must not exist in another branch.

3. For the internal nodes that include a check bit with value *b*, all their descendant internal nodes must have their check bit with a value larger than *b*.

4. From the previous property, PAT tree ensures that any tree path along the PAT tree must contain their check bits in strictly increasing order.

5. With the property of strictly increasing order of tree paths, the sistrings under the same branch share not only a common value at bit-*b*, but also a common prefix up to bit-*b*. It is because when two different sistrings do not have a common prefix up to bit-*b*, it is possible for them to have a common suffix after bit-*b*. In this particular case, we cannot further separate it with any more internal node with check bit larger than *b*. A contradiction occurs consequently.

3.1.4 Some Examples of PAT Tree

Figure 3.6 shows an example of the PAT tree of a simple string '`computer`'. In this example, we decompose it character-by-character and come up with eight sistrings. Table 3.3 shows the bit pattern of each character for a reference. The first 3 bits (bit-0, bit-1 and bit-2) of all the sistrings are the same, and they are `0`, `1` and `1` respectively. The first position with bit difference is at the bit-3, the sistrings '`computer`', '`omputer0`', '`mputer00`', and '`er000000`' contain value `0` and the other four sistrings contain value `1`. Therefore, the root node will have its check bit equals 3. Now we see four sistrings in its left branch and another four sistrings in its right branch. In the left branch, the first position with bit difference is the bit-4 and it further separates two sistrings to the left and two to the right. By further examining its left branch, we note the bit-5 can distinguish between these two sistrings. They finally reach their external node that links directly to the sistring. The other remaining branches break down in a similar way until all of them reach their external nodes. We obtain the final PAT tree configuration as shown in Figure 3.6.

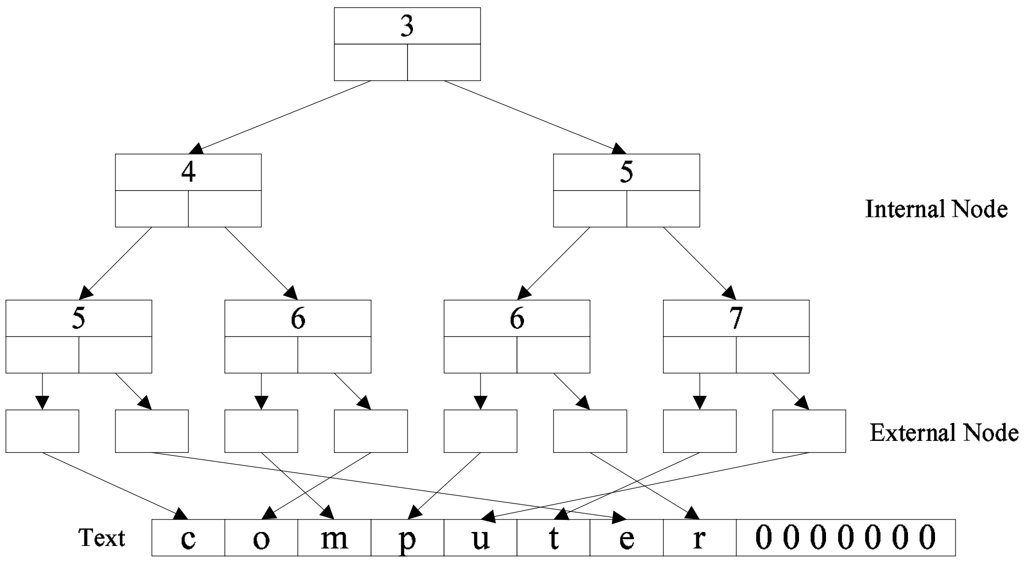


Figure 3.6 Example of PAT tree with the document '`computer`'

Character	**Bit Pattern**
`c`	`01100011`
`o`	`01101111`
`m`	`01101101`
`p`	`01110000`
`u`	`01110101`
`t`	`01110100`
`e`	`01100101`
`r`	`01110010`

Table 3.3 Bit patterns of characters in '`computer`'

Here is another example of PAT tree. We use a simple sentence to illustrate the word-by-word construction of sistrings. The sample text in this example is '`chinese document clustering technique`'. We can express it as four sistrings. We show these four sistrings and their corresponding bit patterns in Table 3.4. The resulting PAT tree contains four external nodes, each for one sistring, and three internal nodes. Figure 3.7 shows the result of the PAT tree in this example.

Sample Text	‘chinese document clustering technique’
Sistring	**Bit Pattern**
chinese document clustering technique	0110001101101000…
document clustering technique000…	$01100\underline{1}0001101111$…
clustering technique000…	$0110001101101\underline{1}00$…
technique000…	$011\underline{1}010001100101$…

Table 3.4 Bit patterns of sistrings in a short sample text

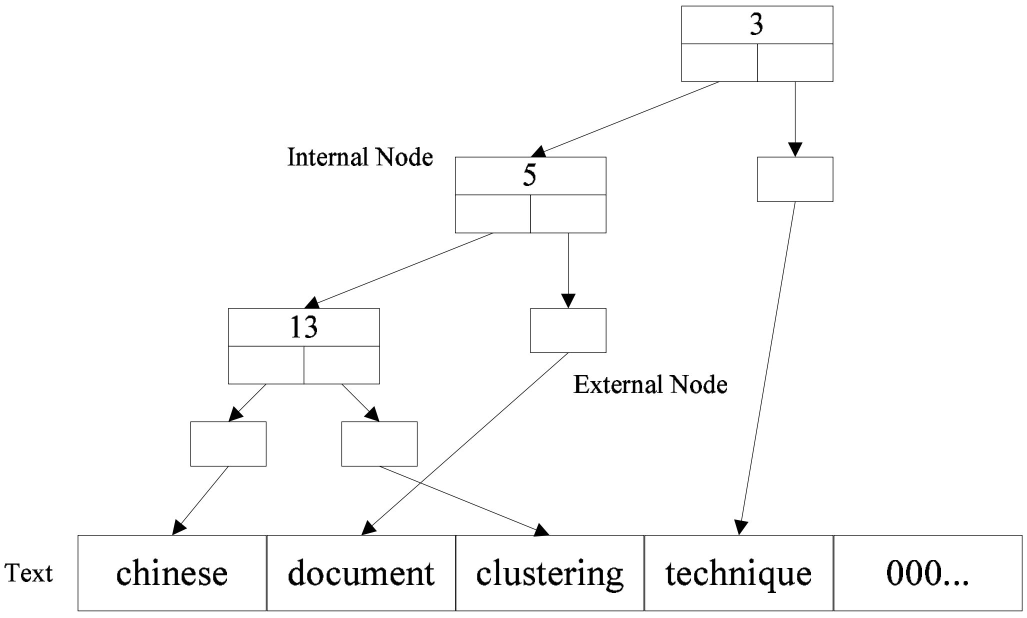


Figure 3.7 Example of PAT tree with the sample text in Table 3.4

3.1.5 Storage Complexity

The storage complexity of PAT tree is linear to the size of the document. We have three kinds of component in PAT tree: external nodes, internal nodes and sistrings. For a document that contains n characters, there are altogether n different sistrings. Since each of the sistrings is pointed by one external node, the storage complexity of the external nodes is $O(n)$. The number of internal nodes in PAT tree would be one less than the number of external nodes because all external nodes must have two children. Therefore, the storage complexity of the internal nodes is $O(n)$ also. For the storage of the sistrings, we could reduce the complexity from $O(n^2)$ to $O(n)$ when we apply the implicit sistrings representation in the original document. The detail of implicit sistrings representation has been discussed in Section 3.1.3.3. It turns out that PAT tree, as a whole, would be in complexity of $O(n)$. Figure 3.8 illustrates the overview of PAT tree.

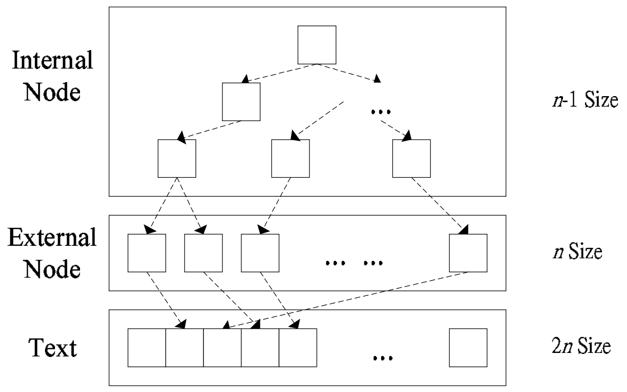


Figure 3.8 Storage overview of PAT tree

The above estimation assumes that PAT tree is built based on characters. In other words, we treat character as the basic element of the document to come up with the resulting sistrings. However, it is not common to interpret a document as group of characters because characters are not a meaningful concept in information processing problems. Usually, we prefer to use words to interpret a document. In this case, words are treated as the basic elements for constructing sistrings. A document with w words will only contain w sistrings, which is an approach more realistic and commonly used. Since $w << n$, the resulting PAT tree becomes more compact, with the storage complexity in $O(n+w)$, which is still bounded by $O(n)$. The overview of this approach is illustrated in Figure 3.9.

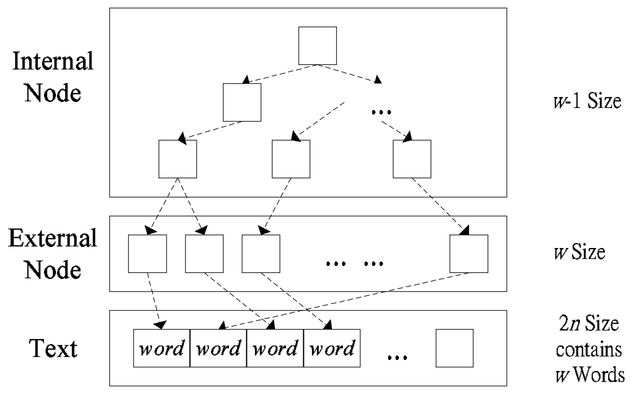


Figure 3.9 Storage overview of PAT tree for documents in word base

3.2 The Chinese PAT Tree

We introduce our variation of Chinese PAT tree in this section. This variation derives from the PAT tree we discuss in the previous section with similar structure layout. The Chinese PAT tree includes the same functionality as the PAT tree for Chinese in [14], and the application on that PAT tree for Chinese can replace the tree with our Chinese PAT tree. The main difference between the two is the node design strategy. Our Chinese PAT tree includes a structure comparable to that of PAT tree but the one in [14] is designed in a reduced-node structure. Comparing the two structures, ours is easier to expend in functionality and more understandable.

In this section, we detail the design and structure of our Chinese PAT tree. Since it is derived from the PAT tree that we discussed in the previous section, we can easily distinguish the difference. We leave the topic about embedded-node design and its problem in the last section of this chapter.

3.2.1 The Chinese PAT Tree Structure

The techniques applied to Chinese are different from those applied to English. Based on the existing techniques for English, we have to, at least, concern about the special property of Chinese and make corresponding changes to cope with the problem due to the language difference. For the PAT Tree, we also need to develop a corresponding Chinese PAT Tree that is suitable for Chinese environment.

3.2.1.1 Chinese Sistrings

"When applying PAT tree on Chinese information processing, instead of recording the semi-infinite strings at document level, we record them at sentence level." [14] This statement clearly points out one of the main differences in the design of Chinese PAT tree. This is due to the nature of Chinese environment. English exhibits the word-by-word sentence structure with each word separated by a space. In Chinese, although there are word boundaries between each sentence, no explicit word boundary can be found within a sentence. This nature is not only the concern in applying PAT tree on Chinese environment, but also the major issue for most information processing techniques in the Chinese environment.

For example, the Chinese sentence "香港中文大學，計算機科學與工程學系" (The Chinese University of Hong Kong, Department of Computer Science and Engineering) includes two phrases separated by a comma. We know that "香港中文大學" is a phrase containing some words while "計算機科學與工程學系" is also a phrase containing some other words. Without any prior knowledge in Chinese languages, we cannot tell what words are inside these phrases.

As we cannot identify words directly in Chinese document, we need to apply the old-fashioned character-by-character formation of sistring in Chinese document. For Chinese PAT tree, we make use of the punctuation information to break down a Chinese document into a bag of Chinese sentences. We can then prepare the list of sistrings from that bag of Chinese sentences. Figure 3.10 shows the example of Chinese sistrings in the sentence level. We assume that each sentence is separated by the punctuation marks. Each of the sentences is padded with null symbols so that all of them are in equal length. Since Chinese characters are double-byte characters, we use the Chinese characters (2-byte) as basic elements of our Chinese sistrings. The first sentence “香港中文大學” that includes 12 bytes (characters) but only six Chinese characters, is represented by six sistrings; the second sentence “計算機科學與工程學系” that includes 20 bytes (characters) but ten Chinese characters, is represented by ten sistrings. As the longest sistring is “計算機科學與工程學系”, which is 20 bytes long, all sistrings would be padded with null symbols to at least 20 bytes.

Chinese Text		“香港中文大學，計算機科學與工程學系”
Sistrings	1	香港中文大學 00000000
	2	港中文大學 0000000000
	3	中文大學 000000000000
	4	文大學 00000000000000
	5	大學 0000000000000000
	6	學 000000000000000000
	7	計算機科學與工程學系
	8	算機科學與工程學系 00
	9	機科學與工程學系 0000
	10	科學與工程學系 000000
	11	學與工程學系 00000000
	12	與工程學系 0000000000
	13	工程學系 000000000000
	14	程學系 00000000000000
	15	學系 0000000000000000
	16	系 000000000000000000

Figure 3.10 Example of Chinese sistrings of a Chinese sentence

3.2.1.2 Problem of Identical Sistrings

Although the sentence level idea for Chinese sistrings suggests a way in handling Chinese document, this approach creates a new problem for the Chinese PAT tree. Chinese sistrings from the sentence level approach can be identical, which does not appear in the original design of sistrings.

When we construct the sistrings in the document level, as mentioned in Section 3.1.2, every sistring in the same document must be of different length before padding any null symbols artificially. Obviously, no matter these sistrings are padded or not, none of them can be identical to another. We always ensure that every sistrings, since they cannot be identical, would definitely locate in an external node with no clashing. However, when the Chinese sistrings in the same document is constructed in the sentence level, before the padding of null symbols, some of them may be of enough length. Obviously, these sistrings with equal length before padding null symbols may be identical. Therefore, we can no longer guarantee that each sistring can locate in one dedicated external node. Namely, collision of nodes is possible to occur in Chinese PAT tree.

Document Text		"創業難，守業更難"
Sistrings	1	創業難 00
	2	業難 0000
	3	**難 000000**
	4	守業更難
	5	業更難 00
	6	更難 0000
	7	**難 000000**

Figure 3.11 Example of Chinese document with identical sistrings

For example, a document with text "創業難，守業更難" produces seven sistring as shown in Figure 3.11, and the third and seventh sistrings are identical with content "難" and padded null symbols. Without a way to distinguish them, node clashing becomes a problem in Chinese documents. In other words, the resulting Chinese PAT tree will index these two sistrings into the same external nodes, which cannot be handled in the PAT tree.

3.2.1.3 Frequency Count

To remedy the problem of identical sistrings that may occur in a Chinese PAT tree, we introduce a new basic element, frequency count, in the external nodes of Chinese PAT tree. The frequency count indicates the number of occurrence of identical sistrings appearing in the document. Figure 3.12 shows the basic structure of an external node of the Chinese PAT tree.

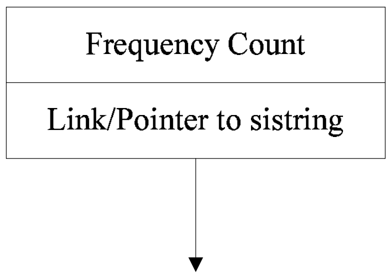


Figure 3.12 Structure of external node of the Chinese PAT tree

Although the resulting Chinese PAT tree will index these two sistrings into the same external nodes, frequency count is a mechanism to capture that kind of event. The frequency count indicates if there are more than one (identical) sistrings referring to a node. For example, in Figure 3.11, the node that links to the character "難" includes a frequency count of two. The pointer of that external node can points to either the third or the seventh sistring as they actually contain the same content.

3.2.2 Some Examples of Chinese PAT Tree

We first use the sample text in Figure 3.10 to illustrate an example of the Chinese PAT tree. The sample text has two sentences and it contains altogether 16 sistrings. The bit pattern of each sistring is shown in Table 3.5. The document breaks down into two null padded sentences and the sistrings are defined in this sentence level. The construction of the Chinese PAT tree is quite similar to the previous PAT tree example and it is shown in Figure 3.13. We can notice that the internal node structure is the same as that of the previous PAT tree, but the external node structure includes an additional frequency count. In this example, no two sistrings are identical and all the frequency counts in the external nodes are equal to one.

The figure can be very complicated when the size of the text is increased.

Although this is a simple example with only 16 external nodes (and 15 internal nodes), the linkage of tree node and sistrings in this example is already quite tedious to show. Therefore, it is not suitable to illustrate the PAT tree figure when the size of document is very large. However, we can still examine the tree structure by this example clearly.

Chinese Text	"香港中文大學，計算機科學與工程學系"
Sistrings	**Bit Pattern**
香港中文大學 00000000	10100100111001011010...
港中文大學 0000000000	10101101101110111011...
中文大學 000000000000	10110100111001001010...
文大學 00000000000000	10100100101001001010...
大學 0000000000000000	10100100011010101011...
學 000000000000000000	10111110110001110000...
計算機科學與工程學系	10101101011100001011...
算機科學與工程學系 00	10111010111000101011...
機科學與工程學系 0000	10111110111101111010...
科學與工程學系 000000	10101100111011001011...
學與工程學系 00000000	10111110110001111011...
與工程學系 0000000000	10111011010100001010...
工程學系 000000000000	10100100011101011011...
程學系 00000000000000	10110101011110111011...
學系 0000000000000000	10111110110001111010...
系 000000000000000000	10101000011101000000...

Table 3.5 Bit patterns of sistrings in the Chinese text of Figure 3.10

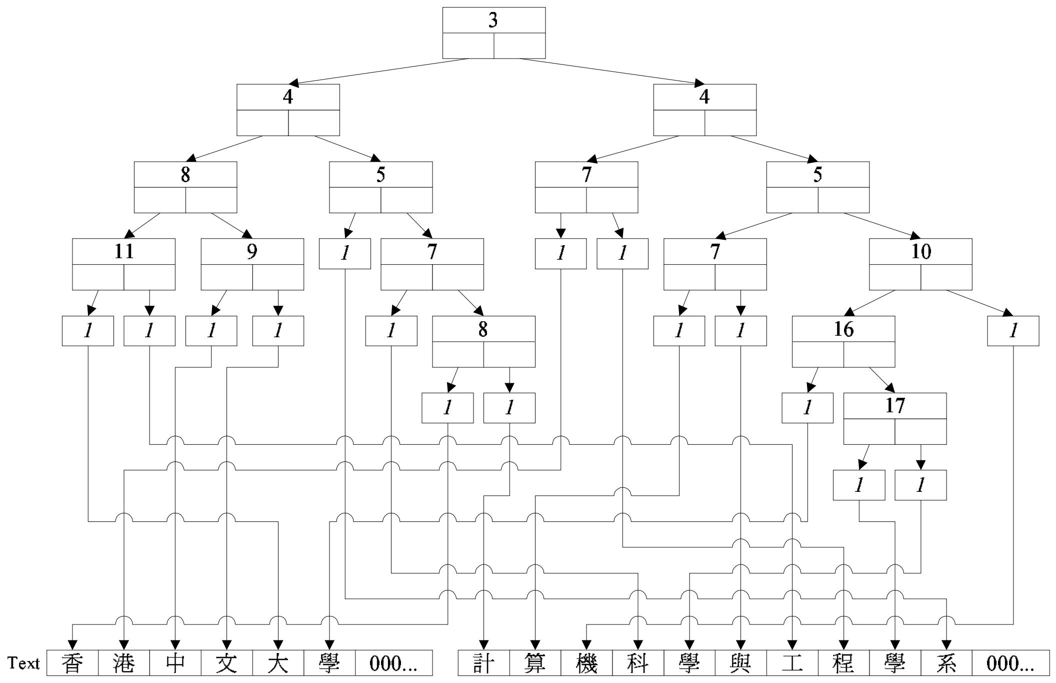


Figure 3.13 Example of Chinese PAT tree with the text in Table 3.5

To demonstrate the usage of frequency count in the external node, we now use the text in Figure 3.11 for another Chinese PAT tree example. This text is with smaller size with an identical sistring appearing twice. The bit pattern of its sistrings is shown in Table 3.6. The corresponding Chinese PAT tree is shown in Figure 3.14. In this example, the tree splits at bit-1, where the left branch continues to split as usual and all the external nodes will point to a unique sistring with frequency count set as one. However, for the right branch, the two sistrings are identical. Both of them are '難 000000' and they will be sharing the same external node with the frequency count set as two. The link to the sistring in that external node can connect to either of them. In Figure 3.14, it is linked to the latter one and therefore, no links would be found from the Chinese PAT tree to the former one. Figure 3.14 also shows a dotted line to link up the identical strings. This is a possible treatment for us to traverse all the identical sistrings. However, depending on application and usage, this dotted line treatment is not always necessary and we will not further discuss it.

Chinese Text	“創業難，守業更難”
Sistrings	**Bit Pattern**
創業難 00	10110$\underline{0}$111101000010...
業難 0000	1011011101111111011...
難 000000	1$\underline{1}$00000111111100000...
守業更難	1010011$\underline{0}$0111010110...
業更難 00	101101110111111101$\underline{0}$...
更難 0000	101$\underline{0}$011111111001111...
難 000000	1$\underline{1}$00000111111100000...

Table 3.6 Bit patterns of sistrings in the Chinese text of Figure 3.11

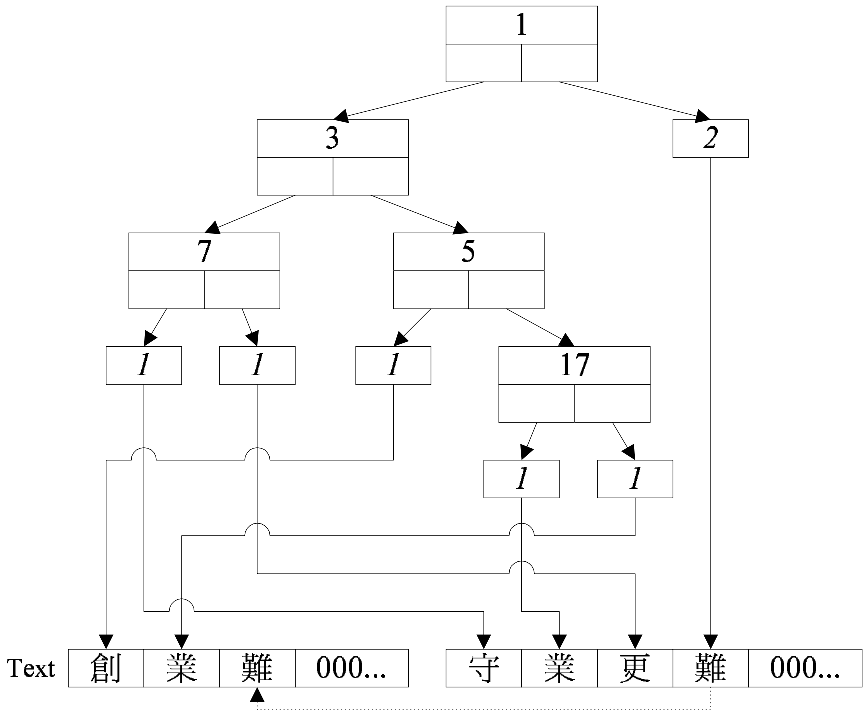


Figure 3.14 Example of Chinese PAT tree with the text in Table 3.6

3.2.3 Storage Complexity

In the examples above, the structure layout of the Chinese PAT tree we proposed here is indifferent with the PAT tree in Section 3.1.5. The Chinese PAT tree for a Chinese document with w word contains maximum of $w+(w-1)$ number of tree nodes, which is bounded by $\mathrm{O}(w)$. The storage of text for sistrings usually double the size of the document, $2w$, which is also bounded by $\mathrm{O}(w)$. Therefore, the storage complexity of the Chinese PAT tree is the same as in Figure 3.9, which is a linear storage structure.

3.3 The Generic Chinese PAT Tree

We propose a variation of Chinese PAT tree, which is derived from the Chinese PAT tree that we discuss before. Although the PAT tree for Chinese is widely used in Chinese information processing, its structure contains some weaknesses. Our generic Chinese PAT tree is extended from the Chinese PAT tree structure in the previous section that features the basic Chinese capabilities. The generic Chinese PAT tree enhances over the Chinese PAT tree so that it is more appropriate for document clustering. Besides, the generic Chinese PAT tree helps simplifying the clustering process and its structure is clear to understand.

3.3.1 Structure Overview

In the generic Chinese PAT tree, we do not distinguish the internal nodes and external nodes explicitly. They share the same uniformed node structure. As we know, there are four basic elements in the node of PAT tree: the check bit and child pointers are the elements in the internal nodes while the link to sistrings and the frequency count are the elements in the external nodes. All the tree nodes in the generic Chinese PAT tree would include all these four basic elements. We further extend the meaning and functionality of each of these basic elements such as the meaning of check bit in the leaf node.

By eliminating the explicit boundary of internal and external nodes, we introduce a new category of node called *Essential Node* (*EN*). It is a node with critical information related to the clustering process. While the essential node contains the clustering information, the non-essential node solely serves as the index to maintain the PAT tree structure. With this modification, we can simplify our clustering process with the aid of this generic Chinese PAT tree.

3.3.2 Structure of Tree Nodes

In our generic Chinese PAT tree structure, each of the tree node consists of the following components: (1) a check bit, (2) a link to a sistring, (3) a frequency count of the phrase which is represented by the current node, and (4) left and right pointers to the child nodes. Here we describe the details of each of these components in our generic Chinese PAT tree. An example of the structure is illustrated in Figure 3.15.

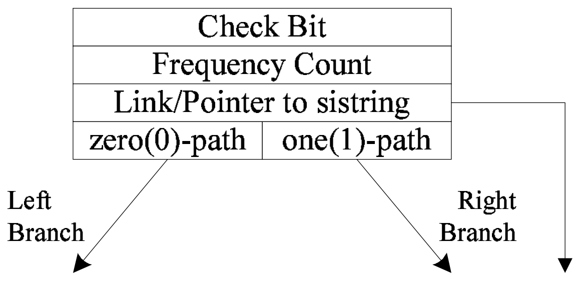


Figure 3.15 Structure of tree node of the generic Chinese PAT tree

3.3.2.1 Check Bit

Check bit is the key information of the internal nodes in the original PAT tree. It indicates the first bit of the underlying sistrings that differentiate them into two groups, one storing in the left sub-trees while another group storing in the right sub-trees. The branching decision in the internal nodes depends on this check bit.

The check bit is introduced in the leaf node of our generic Chinese PAT tree. Although branching decision is not required in the leaf, we still include the check bit for the reason of consistency. We define the check bit of the leaf node to be equal to the bit length of the associated sistring. For example, if the node is a leaf node linking to the sistring ‘電腦 000…’, the sistring will have two Chinese characters occupying 4-byte length. Without considering the padding null symbols, it contains 32 bits of useful content. So the check bit would be equal to 32.

3.3.2.2 Link to a Sistring

For the leaf nodes, this link can identify the sistrings uniquely. Each sistring can only appear in exactly one leaf node. For example, ‘香港體育館’ is a string consisting five sistrings: ‘香港體育館’, ‘港體育館 00’, ‘體育館 0000’, ‘育館 000000’ and ‘館 00000000’. If we construct the generic Chinese PAT tree with these sistrings, we will produce exactly five leaf nodes with each of them linked to one of the sistrings. Another example, ‘香港好，中國好’ is a string consisting six sistrings: ‘香港好’, ‘港好 00, ‘好 0000’, ‘中國好’, ‘國好 00’ and ‘好 0000’. Since there are two duplicated sistrings ‘好 0000’, there are only five unique sistrings. In the corresponding generic Chinese PAT tree, we will see five leaf nodes and the link to the sistring ‘好 0000’ will appear once in one of the leaf node. That leaf node may link to either the former or the latter sistring ‘好 0000’.

The link to sistring in the non-leaf nodes is newly introduced in the generic Chinese PAT tree. We include this link information to maintain the consistency of the tree node. By definition, the link will point to the sistring where any of its children is pointing. Since any non-leaf node must include exactly two children, for simplicity, we can assume that the link will always be the same as the one of its left child. We notice that all these additional links are redundant because all the sistring links in the non-leaf nodes are referring to one of its children.

Although the modification introduces some redundancy on sistring links, we do provide new information for the non-leaf node. The traditional PAT tree structure includes its internal nodes as the index and its external nodes as sistrings information. Our modification treats these nodes equally, each of them will contain its check bit as well as the link to a sistring. While the leaf node will represent the whole sistring it belongs to, the non-leaf node may also represent a part (prefix) of a sistring up to the check bit position. Therefore, the generic Chinese PAT tree does retain the indexing structure and it also enriches the information for each tree node. The details about the relation of the representation to the check bit and sistring link are formalized in the definition of *Essential Node* (*EN*) to be discussed later.

3.3.2.3 Frequency Count

The frequency count indicates the frequency of the corresponding character string (Chinese phrase) that the node can represent. For the leaf nodes, it is simply the occurrence of their associated sistring, and for the non-leaf nodes, we define it as the sum of the frequency counts of their left child and right child.

3.3.2.4 Pointers to the Child Nodes

Each non-leaf node must include pointers to both left subtree and right subtree. The left subtree contains tree nodes with sistrings of value 0 in the check bit position, and the right subtree, on the other hand, contains tree nodes with sistrings of value 1 in the check bit position. The definition is the same as the traditional PAT tree.

For the leaf nodes, they do not contain any child nodes and should not include such pointers. Again, we retain such information in the leaf nodes mainly for consistency. By convention, their child pointers are always pointing to *null*. This can implicitly indicate that the tree node is a leaf node.

3.3.3 Essential Node

Essential Nodes (*EN*) can provide key information in the generic Chinese PAT tree. It is, basically, PAT tree nodes that can represent a Chinese phrase in a meaningful way. With Essential Nodes, we can apply further techniques and analyses on the PAT tree, based on the Essential Node information, correctly and efficiently. We first give definition of *Essential Length*. Based on the definition of Essential Length, we further define the Essential Node in our generic Chinese PAT tree.

3.3.3.1 Essential Length

In the generic Chinese PAT tree, we define the essential length of node *x*, *Essential Length*$_x$, to be equal to the check bit of *x* truncated to the nearest Chinese character (16 bits). In another words,

$$Essential\ Length_x = check\ bit_x - (check\ bit_x \bmod 16)$$

Essential Length is a quantitative measure of how much Chinese information a tree node can represent. As each of the tree nodes in the generic Chinese PAT tree would link to a sistring, each of them may represent a prefix of that sistring. When the tree node is a leaf node, it will represent the whole sistring it is linked to. Although the check bit is useful in providing the information of the number of bits the tree node can represent, it is only in the bit level. As we are interested in the number of Chinese characters, not the number of bits, the tree node can represent, we do count the check bit in a 16-bit step, which is the bit size of a single Chinese character (in Big-5 encoding).

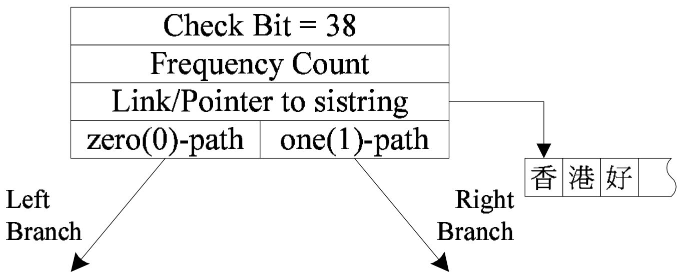


Figure 3.16 Example of tree node of the generic Chinese PAT tree

For example, a tree node shown in Figure 3.16 contains a pointer that links to the sistring ‘香港好’ with a check bit equals 38. This indicates that the tree node may represent the first 38 bits of the sistrings ‘香港好’ and all its children will also include the first 38 bits (bit-0 to bit-37) in common. Although this tree node can represent altogether 38 bits of the sistrings, it cannot represent the whole sistring ‘香港好’, which is 48 bits. However, it can represent the first two Chinese characters, ‘香港’, which is 32 bits. The essential length of the node in Figure 3.16 would be $38-(38 \bmod 16)=38-6=32$.

The above Essential Length definition is a particular case of Chinese characters. We assume that Chinese characters are double-bytes characters, which are each 16-bit long. While we are referring these Chinese characters in Big-5 encoding scheme, it is the same for GB encoding scheme. However, the above definition does not hold in all encoding schemes. For example, in UNICODE encoding scheme, Chinese characters are 32-bit long, and we need to revise the definition into,

$$Essential\ Length_x = check\ bit_x - (check\ bit_x \bmod 32)$$

In fact, we can generalize the definition of Essential Length into,

$$Essential\ Length_x = check\ bit_x - (check\ bit_x \bmod bit\text{-}size)$$

where $bit\text{-}size$ is the number of bits for a Chinese character.

For documents in Big-5 or GB encoding scheme, *bit-size* for each Chinese character is 16; and for documents in UNICODE, *bit-size* for each Chinese character is 32. If there is a new Chinese encoding scheme with *bit-size* of 64 or otherwise, we can still use our generalized definition to determine the Essential Length. For simplicity, our study assumes that we use Big-5 encoding scheme so the *bit-size* is always 16.

3.3.3.2 Essential Node

In the generic Chinese PAT tree, a node (x) is an *Essential Node* (*EN*) if and only if

(1) $\textit{Essential Length}_x \geq 2 \times \textit{bit-size}$; and

(2) $\textit{Essential Length}_x - \textit{Essential Length}_y \geq \textit{bit-size}$ (if such y exists),

where node (y) is the nearest ancestor of x such that y is also an *EN*.

In particular, we assume that *bit-size* of Chinese documents is always 16, so we have our restricted definition for simplicity. The restricted definition tells that a node (x) is an *Essential Node* (*EN*) if and only if

(1) $\textit{Essential Length}_x \geq 32$; and

(2) $\textit{Essential Length}_x - \textit{Essential Length}_y \geq 16$ (if such y exists),

where node (y) is the nearest ancestor of x such that y is also an *EN*.

The above definition ensures that each essential node represents a Chinese phrase and no two essential nodes overlap to represent the same Chinese phrase. When there is an essential node (y), which is an ancestor of x, we compare their Essential Lengths. If their Essential Lengths differ in less than 16, that means x and y are representing exactly the same Chinese phrase. Therefore, x cannot be an Essential Node. On the other hand, if their Essential Lengths differ in at least 16, that mean x can represent a Chinese phrase different from y. Therefore, x is an Essential Node.

However, when x has no such ancestor y, the second rule is not applicable. In that case, the first statement of the definition provides the minimum criteria of an *Essential Node*. A node with Essential Length less than 16 can represent no Chinese character, and a node with Essential Length less than 32 can represent no more than one Chinese characters. Since a Chinese phrase should be composed of at least two Chinese characters, x must contain its Essential Length of at least 32 to be an *Essential Node*, meaning that x represents a phrase of at least two Chinese characters.

In the generic Chinese PAT tree, we can determine Essential Nodes with a top-down decision approach. We consider the structure of the generic Chinese PAT tree from the root. By locating the top most *Essential Nodes* from the first rule, we can derive the remaining *Essential Nodes* along the tree path. The next *Essential Node* must include its Essential Length greater than its ancestral *Essential Node* by at least 16. In other words, the next one would represent at least one more Chinese character than the previous one. The process requires only one parse of the generic Chinese PAT tree. No revisit of nodes is required.

From the *Essential Node* definition, each *Essential Node* can represent a Chinese phrase, *p*, which equals to its linked sistring from bit-0 up to the $\mathit{Essential\ Length} - 1$. Each *p* must contain at least two Chinese characters (32-bit) and is represented by at most one *Essential Node*. Figure 3.17 and Figure 3.18 are examples of a generic Chinese PAT tree with their *Essential Nodes* being highlighted (their check bit is shaded). These figures will be described in details in the following section.

3.3.4 Some Examples of the Generic Chinese PAT Tree

Here we show two examples of the generic Chinese PAT tree. In these examples, we notice a major difference in generic Chinese PAT tree, as all nodes are in a unified structure. The first one is a simple example without identical sistrings and the second one contains a pair of identical sistrings inside.

In the first example, we construct the generic Chinese PAT tree of the Chinese Text “香港體育館”. The bit pattern of its sistrings is shown in Table 3.7. The first bit-difference among these five sistrings is at bit-1, where three sistrings have value 0 and two sistrings have value 1. Therefore, the root node of the tree include check bit equal to 1 and frequency count equal to 5. For the left branch, comparing bit-3 further separates the sistrings. The following right branch includes a remaining sistring and it reaches the leaf node. The tree pointers in the leaf node are kept null, which is an implicit signal to denote the leaf node. The check bit in the leaf node is equal to 64, which is an artificial value that equals to the bit length of the sistring by omitting the null padded symbols. As we mentioned before, the link to the sistring in the non-leaf nodes is a redundant link and we will assume that it is the same as the link in its left branch. This example also demonstrates this property and the final tree configuration is shown in Figure 3.17.

Chinese Text	"香港體育館"
Sistrings	**Bit Pattern**
香港體育館	10101101...
港體育館 00	10110100...
體育館 0000	11000101...
育館 000000	10101000...
館 00000000	11000000...

Table 3.7 Bit patterns of sistrings in the Chinese text: "香港體育館"

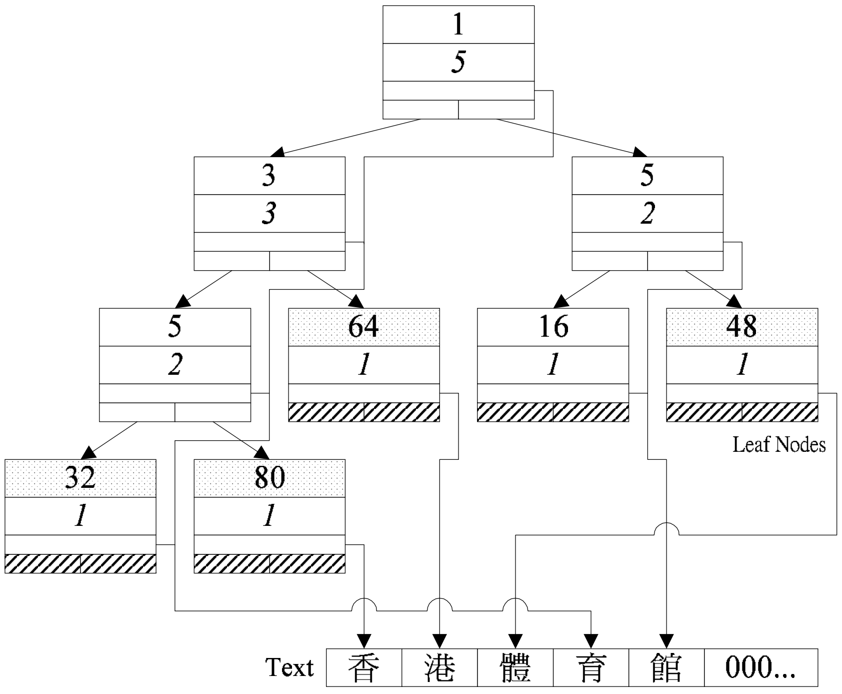


Figure 3.17 Generic Chinese PAT tree with the text in Table 3.7

The second example illustrates the generic Chinese PAT tree of a Chinese text that contains identical sistrings. We use a short text “香港好，中國好” in this example. You can denote that a sistring “好 000…” appears twice and it is the identical sistrings pair in this example. Table 3.8 shows the bit pattern of the Chinese text and the corresponding generic Chinese PAT tree is presented in Figure 3.18. The tree node with a check bit equal to 16 represents the sistring “好 000…”. Since this sistring appears twice in the Chinese text, that tree node is recorded with frequency count equal to two and the link to the sistrings will connect to either of these two sistrings. In this example, it is connected to the latter one.

From this example, we notice that although the complexity of PAT tree is bounded by $O(n)$, the actual memory requirement may be reduced to a certain extent. The original Chinese text contains six Chinese characters, which is of 12 bytes size (i.e. n = 12). The memory required for the sistring is, however, less than 2n size and it requires only 20 bytes (including padded symbol). The number of leaf nodes usually equals the number of Chinese characters, which is $0.5 \times n = 6$. Due to the existence of identical sistrings, the actual number of leaf nodes is one less than this number (i.e. five leaf nodes) and the actual number of non-leaf nodes is four leaf nodes, which is the number of leaf nodes minus one. Therefore, the actual memory consumption in generic Chinese PAT tree could be reduced depending on the Chinese text with a known upper bound.

Chinese Text	“香港好，中國好”
Sistrings	**Bit Pattern**
香港好	1010110110111011…
港好 00	1011010011100100…
好 0000	1010011001101110…
中國好	1010010010100100…
國好 00	1011000011101010…
好 0000	1010011001101110…

Table 3.8 Bit patterns of sistrings in the Chinese text: “香港好，中國好”

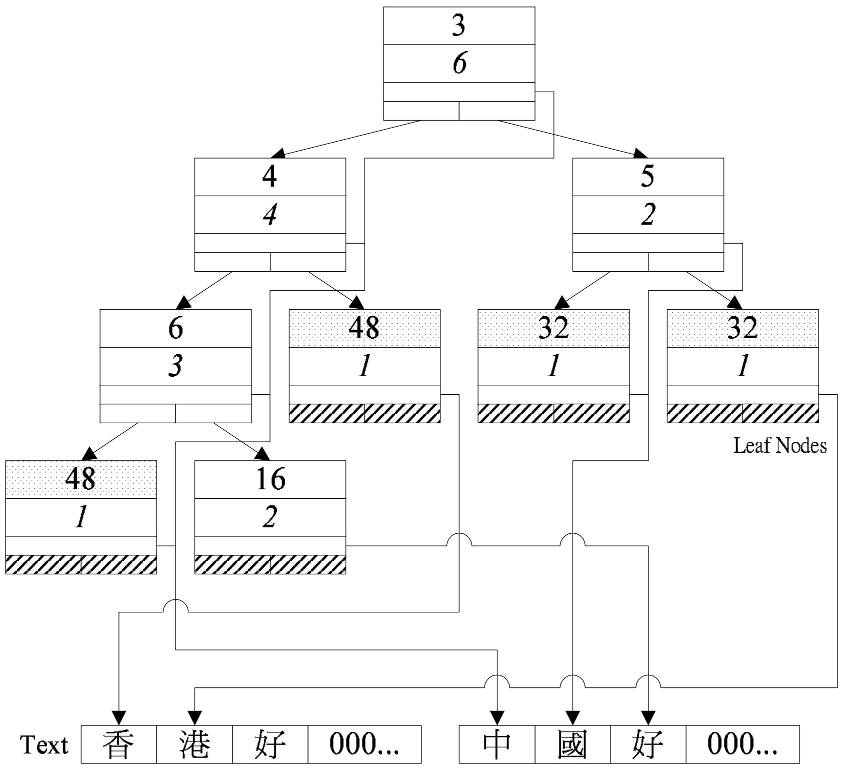


Figure 3.18 Generic Chinese PAT tree with the text in Table 3.8

The nodes with their check bits shaded in Figure 3.17 and Figure 3.18 are *Essential Node*. From these examples, we notice that each *Essential Node* would represent a Chinese phrase. Although the *Essential Nodes* in the previous examples are all leaves, *Essential Nodes* may also appear in the non-leaf nodes. The next generic Chinese PAT tree example will illustrate the case where the *Essential Nodes* are in the non-leaf nodes. With the Chinese Text in Table 3.9, we can construct the generic Chinese PAT tree as shown in Figure 3.19. This tree includes ten leaf nodes while eight of them are *Essential Nodes*. For the non-leaf nodes, two of them, one with check bit of value 53 and another with check bit of value 37, satisfy the definition of Essential Node. The tree altogether includes ten Essential Nodes inside, each representing the linked sistring up to its *Essential Length*.

Chinese Text	“計算機科學，計算機系統”
Sistrings	**Bit Pattern**
計算機科學	10101101011100001011101011100010101111...011...
算機科學 00	10111010111000101011111011111011111011011...111...
機科學 0000	10111111011111011111010110011101100101111...000...
科學 000000	10101100111011001011111101100011100000000...000...
學 00000000	10111111011000111000000000000000000000000...000...
計算機系統	10101101011100001011101011100010101111...010...
算機系統 00	10111010111000101011111011111011111010110...100...
機系統 0000	10111111011111011111010100001110100101100...000...
系統 000000	10101000011101001011001011001110000000...000...
統 00000000	10110010110011100000000000000000000000000...000...

Table 3.9 Bit patterns of sistrings in the Chinese text: “計算機科學，計算機系統”

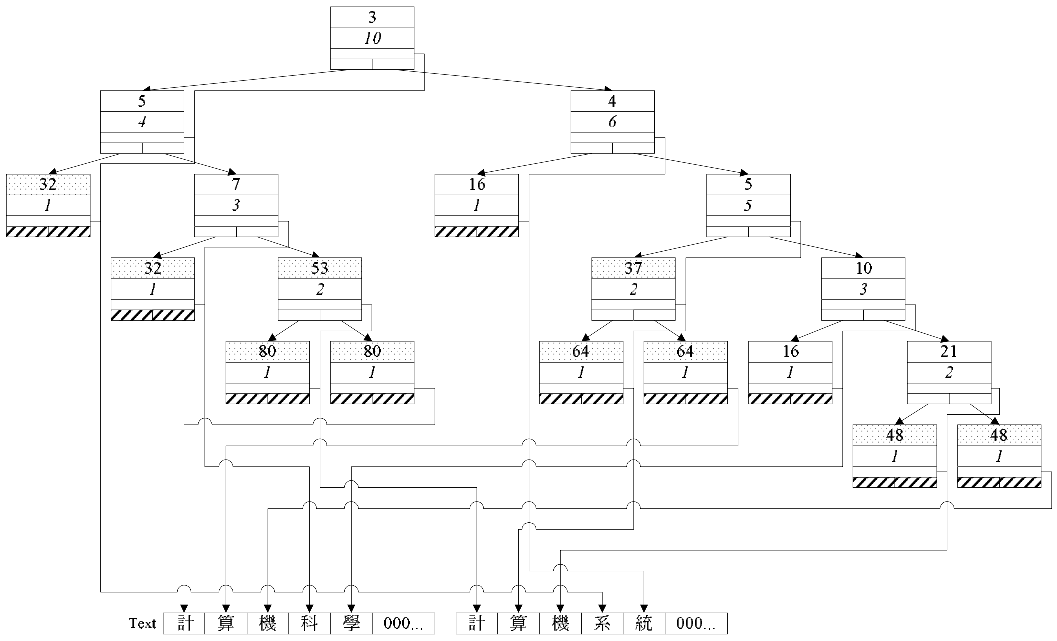


Figure 3.19 Generic Chinese PAT tree with the text in Table 3.9

3.3.5 Storage Complexity

From the different examples shown above, we notice a number of changes in the generic Chinese PAT tree that we propose. Although our generic Chinese PAT tree has extended functions and abilities that make the node structure look more sophisticated, our changes do not deform the structure of the PAT tree. Since our enhancements are about the internal structure of a tree node as well as its usage and interpretations. These changes do not affect the storage complexity of the PAT tree. Besides, identical Chinese phrases are located in the same node, so that the number of tree nodes is usually much less than this expected maximum bound in practice. Therefore, the complexity analysis of the PAT tree in Section 3.1.5 and the Chinese PAT tree in Section 3.2.3 still hold. For a document with w words, the storage complexity of the generic Chinese PAT tree is $O(w)$, which is a linear data structure.

In fact, our size of the tree node in the generic Chinese PAT tree is slightly larger. Extra memory is required to maintain the generalized structure as well as the information of Essential Length and Essential Node. However, this drawback is insignificant due to the advancement of computer technology. The cost of computer memory is dropped dramatically over 60 percent in five years of time. At the same time, modern computers are capable to equip large amount of memory. A personal computer (PC) can easily contain gigabytes of memory. The extra memory consumption of tree node is minor drawback to our generic Chinese PAT tree structure design; however, the generic Chinese PAT tree still maintain the data structure in linear storage complexity.

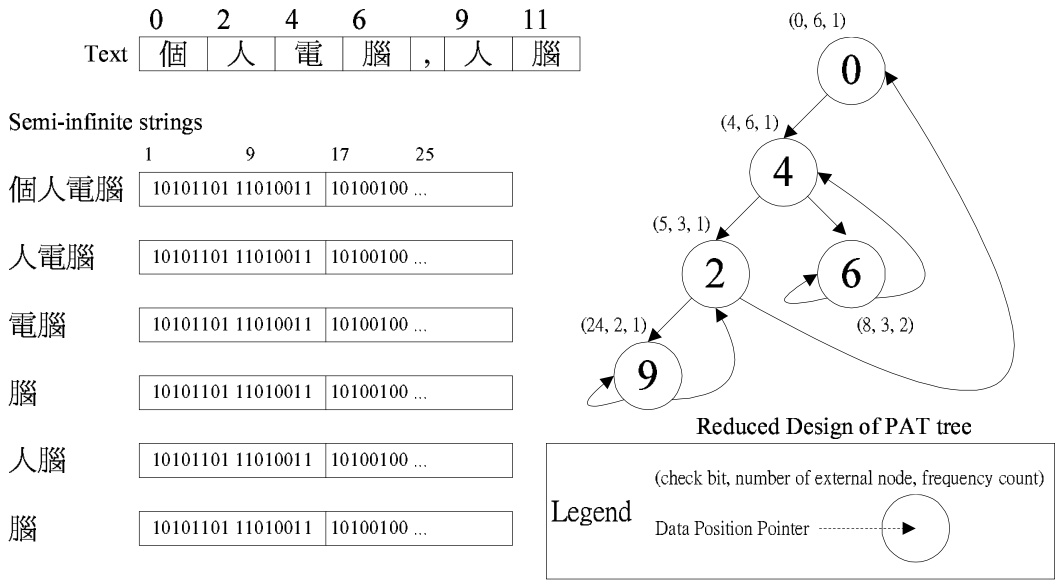
3.4 Problems of Embedded Nodes

In this section, we illustrate the weaknesses or limitations of embedded nodes design for the PAT tree data structure for Chinese. This embedded node design is the way of handling Chinese information in the PAT tree that was proposed in [14]. The reduced structure embeds the internal nodes with external nodes, which reduce the actual number of nodes. Although it is an elegant way that tries to reduce the tree size, the reduced structure does not reduce much of the memory consumption. At the same time, we notice a number of weaknesses and disadvantages in the reduced structure of PAT tree. This section discusses this embedded node design in [14] and the disadvantages of the reduced structure.

3.4.1 The Reduced Structure

Figure 3.20 is an example that shows the PAT tree for Chinese in reduced structure with the typical example text, "個人電腦,人腦". This structure is a reduced structure to combine the leaf nodes with an internal node, so that the number of nodes in the tree is reduced by about half. As we have seen, the node structure is more complicated and difficult to understand because the nodes are over-loaded with information mixed together. Here is the interpretation of the structure: that tree contains no leaf nodes physically, and the nodes we can see in the structure are internal nodes. From the definition, a PAT tree with n (identical) sistrings should include n leaf nodes and $n-1$ internal nodes. So the example in Figure 3.20 includes five identical sistrings, and there should be five leaf nodes and four internal nodes. The structure of the embedded design attempts to combine a leaf node with an internal node. By doing so it artificially creates one more internal node above the tree root. Therefore, in the reduced design, each internal node represents one and only one leaf node at the same time, depending on its context.

In Figure 3.20, Node 4 was the root node of the PAT tree, but there is an artificial node, Node 0, that is created on top of Node 4. The tree contains totally five nodes. They form the internal nodes of the PAT tree with their corresponding check bit information. Since the PAT tree requires that the check bit information along the tree path must be strictly increasing, this becomes the hints to determine whether the accessed node serves as an internal node or not. When the tree reaches a node without an increase on check-bit, we know that it has reached its leaf node. In Figure 3.20, all the links pointing back to themselves or any other ancestral nodes are the links to leaf nodes.



0 2 4 6 9 11
Text 個 人 電 腦 ， 人 腦
Semi-infinite strings
1 9 17 25
個人電腦 10101101 11010011 10100100 ...
人電腦 10101101 11010011 10100100 ...
電腦 10101101 11010011 10100100 ...
腦 10101101 11010011 10100100 ...
人腦 10101101 11010011 10100100 ...
腦 10101101 11010011 10100100 ...
(0, 6, 1)
0
(4, 6, 1)
4
(5, 3, 1)
2
6
(24, 2, 1)
(8, 3, 2)
9
Reduced Design of PAT tree
Legend
(check bit, number of external node, frequency count)
Data Position Pointer

Figure 3.20 Typical example of PAT tree in reduced node design

3.4.2 Disadvantages of Reduced Structure

3.4.2.1 Violate the PATRICIA Structure

As we know that PAT tree is a kind of PATRICIA tree, each of the internal nodes of the PAT tree must have exactly two branches to their children. However, in the embedded node design, the reduced structure does not contain enough internal nodes to combine with those leaf nodes. As a result, the artificial node is created at the top of the tree and it becomes the root node. Unfortunately, that artificial node only includes has one child, the original root of the PAT tree. This violates the assumption of the PAT tree and the PATRICIA tree. Although this violation does not harm the functionality of PAT tree, it makes the tree structure more difficult to maintain. Special treatment is needed and the structure becomes very specific.

3.4.2.2 Ambiguity

Regardless of the existence of the artificial internal node, the reduced structure is still ambiguous. The PAT tree in the reduced structure is ambiguous in the information they are storing. By inspecting a tree node, the information is all there but we never know it is for internal node, for leaf node or actually for both. Usually, the check bit information is the key for the internal node, but it also acts as an indicator to detect a “leaf node” in the reduced structure. The frequency count and the data position pointer are actually information for the leaf node. The backward tree link at the end of every branch makes the tree structure even more complicated.

In fact, the reduced structure can only reduce the number of nodes by about half, but it does not reduce the memory requirement of the data structure. By means of memory storage, the combination of internal nodes with leaf nodes still requires the same amount of memory for the check bit, pointer links, frequency count, etc. By means of big-O storage complexity, although the reduced structure eliminates half of the number nodes, it does not improve the complexity any further, and the overall is still a $\mathrm{O}(n)$ structure.

3.4.2.3 Limiting Expansion

In the reduced structure, it is very complicated, if not unlikely, to make improvement on the functionality of the tree. One of the important changes in our generic Chinese PAT tree is to utilize the tree nodes such that every node is capable of maintaining useful information, so we can enhance the overall functionality through the formal definition of essential node properties and expendable node information capability. However, in the reduced structure design, the information in a single node is very confusing in storing the same type of information to represent the different types of node simultaneously. This design is undesirable and it limits the expansion that we require.

3.4.3 A Case Study of Reduced Design

To illustrate the problems and confusion in the reduced structure design, we conduct a case study that refers to the example of PAT tree in the reduced design in Figure 3.20. We demonstrate the effect of the reduced structure design that can cause confusion or complicated interpretation.

Since the reduced structure of the PAT tree combines the leaf node with an internal node, it only retains information for the sistrings on the tree. In other words, all strings that are not sistrings cannot retain the same kind of information as in the sistring. For example, "電腦" is a sistring, so we can retrieve the phrase "電腦" from the tree and its node (Node 4) tells us that the frequency of "電腦" is 1. However, "個人" is not a sistring. When we try retrieve it from the tree, we move down the tree and reach the node "個人電腦" (Node 0) only. We may not know the correct frequency of "個人". Although we still obtain the frequency 1 by inspecting the node "個人電腦", which is correct frequency for "個人" as well, that reported frequency is not necessarily the correct frequency for all strings. When we consider "人", which should have frequency 2, in this example, we finally reach the node "人腦" (Node 9) and only obtain the frequency 1 from that node. This is because the node "人腦" is containing frequency of "人腦", which is 1, but not the frequency of "人".

In fact, the correct frequency count of the phrase "人" is still maintained in this reduced structure of PAT tree, which is indicated as the "number of external node" in Node 9. Since a node is representing a leaf node and an internal node at the same time, the correct frequency of phrase "人" is in the "number of external node", which is part of the internal node information. As we can see, although it is not totally impossible for reporting the frequency count, the way to maintain the correct and persistent information becomes very tricky and non-uniform.

This case study uses the frequency count to illustrate how confusing and complicated the reduced design is. When the tree requires containing a more sophisticated value, which is not just a matter of frequency counting, the problems of the reduced structure design can be more critical.

3.4.4 Experiments on Frequency Mismatch

The case study is an example that illustrates the potential problem of the reduced structure. With the assumption that one kind of information is in the leaf nodes, the reduced structure cannot retain that information in the internal node. Since the leaf nodes and internal nodes are equally important to represent Chinese phrase information, the reduced structure confuses the information on a tree node. To quantify the mismatch caused by this problem, we conduct experiments that use a simple value in the tree node, frequency count, to illustrate the potential problems and difficulties in the reduced structure of PAT tree.

3.4.4.1 Analysis using Phrase Strings

We analyze the number of phrasal information of a Chinese document and measure the percentage of mismatch reported directly from the reduced structure of PAT tree. For example, The number of phrases in “個人電腦” is ten: “個”, “人”, “電”, “腦”, “個人”, “人電”, “電腦”, “個人電”, “人電腦”, “個人電腦”. We query the reduced structure of PAT tree, examine the frequency count of that associated node, and record the number of mismatch of frequency count to the actual frequency of phrase.

	Number of Phrases	Reported Mismatch	Percentage (%)
1.	4584	372	8.12%
2.	3372	126	3.74%
3.	4096	200	4.88%
4.	2539	146	5.75%
5.	1171	67	5.72%
6.	4894	274	5.60%
7.	4252	255	6.00%
8.	1666	116	6.96%
9.	4411	254	5.76%
10.	4007	208	5.19%
Total	34992	2018	5.77%

Table 3.10 Analysis of phrase strings in reduced structure of PAT tree

The experiment repeats ten times on different articles. Table 3.10 shows the result of the experiment. Among the samples, the percentage of reported mismatch is between 3.74% and 8.12%. On average, there is about 5.77% mismatch of frequency report for all possible phrases string.

Since phrase string is a well-defined elements. The measurement in this experiment is objective. It can reflect the effect of reduced structure in terms of phrases information.

3.4.4.2 Analysis using Phrase Nodes

This experiment uses the same setup of ten documents to analyze the effect of the reduced structure in terms of tree nodes. Since a node may represent more than one phrase string, depending on the actual content of the document, the experiment result in Section 3.4.4.1 includes the over-counts of PAT tree nodes. To obtain the similar figure in terms of nodes, our second experiment counts the number of tree nodes that should represent Chinese strings, and report the mismatch among these nodes. The mismatch may happen when a string is expected to locate in an 'internal node' and the frequency value of that node, which should be an information for the leaf node, does not match with the string. Table 3.11 presents the measurement of this observation.

	Number of Nodes	Reported Mismatch	Percentage (%)
1.	847	136	16.06%
2.	442	34	7.69%
3.	616	59	9.58%
4.	359	41	11.42%
5.	220	18	8.18%
6.	779	86	11.04%
7.	610	76	12.46%
8.	282	38	13.48%
9.	647	82	12.67%
10.	599	70	11.69%
Total	5401	640	11.85%

Table 3.11 Analysis of phrase nodes in reduced structure of PAT tree

From the result, the percentage of the reported mismatch is between 8.18% and 16.06%. It is 11.85% on average. Compared with the result in Table 3.10, the percentage of affected nodes is more than the percentage of affected strings. We can expect this difference in the result because most of the strings are represented by leaf node and they always contribute a correct frequency count in the first experiment. The second experiment evaluates in terms of nodes, which are in reduced structure so that an internal node and a leaf node are embedded into a single node. Consequently we miss the correct frequency count due to 'leaf nodes'. This experiment is also based on an objective measurement in the perspective of tree nodes.

3.4.4.3 Analysis using Chinese Vocabularies

The third experiment employs the same set of documents in terms of Chinese vocabularies. Chinese vocabularies are human sensible words that contain particular meanings in Chinese language. The analyses by using words are more meaningful. It can reflect application needs in general.

The experiment engaging Chinese vocabularies is more complicated than the previous two. We identify words from each of the ten test documents manually to determine the number of words in the document. The count includes simple vocabularies with dedicated meanings, for example, "個人電腦" contains two words, which are "個人" and "電腦". Only the well-defined words are included in this measurement.

	Number of Words	**Reported Mismatch**	**Percentage (%)**
1.	200	65	32.50%
2.	106	22	20.75%
3.	154	37	24.03%
4.	87	20	22.99%
5.	67	11	16.42%
6.	167	41	24.55%
7.	138	34	24.64%
8.	67	15	22.39%
9.	137	35	25.55%
10.	146	33	22.60%
Total	1269	313	24.67%

Table 3.12 Analysis of Chinese words in reduced structure of PAT tree

The result in this experiment analysis shows that meaningful Chinese vocabularies are likely to retain in the internal node and their actual frequency report to mismatch with the frequency count value. In the test set, the percentage of mismatch is between 16.42% and 32.50%. It is 24.67% on average.

This experiment result is a subjective measurement because the definition of Chinese vocabularies does not have a strict boundary. However, with the reference to the previous two objective measurements, we have the overall ideas on how the reduced structure can affect the performance of the PAT tree. It becomes a critical problem especially when we are dealing with Chinese words.

3.5 Strengths of the Generic Chinese PAT Tree

The generic Chinese PAT tree contains a number of differences from the one in the reduced design. It also improves in functionality over the Chinese PAT tree. Without affecting the $O(n)$ storage complexity, the generic Chinese PAT tree has the following strengths:

First, the node structure in the generic Chinese PAT tree is uniform. There is no physical difference between the internal node and the external node. We avoid the heterogeneous structures in handling the internal node and the external node separately. This advantage makes our generic Chinese PAT tree more understandable and easier to maintain.

Secondly, the generic Chinese PAT tree nodes are rich in information. Each of the generic Chinese PAT tree nodes has all four basic components: check bit, frequency count, link to sistring, and child pointers. By extending the definition of each of these components in the original data structure, we make the tree nodes capable of keeping information on each node. The enriched node content, especially in the internal nodes, can contain meaningful information. That can also improve the expressiveness of the tree.

Thirdly, the generic Chinese PAT tree node has the new attribute called *Essential Length*, which is a dependent value of the check bit. This *Essential Length* provides important information for the tree related to the Chinese document. It is a higher-level description that reflects the expressiveness of Chinese phrases in a node. It also determines if the node is an *Essential Node*.

Finally, we introduce a special type of node called *Essential Node*, which is a node in the generic Chinese PAT tree that can represent a Chinese phrase (a series of Chinese characters) properly. *Essential Node* is a node property from the

Essential Length information as well as the tree structure. We can only obtain that information after we unify the external node and internal node into a common node structure. The definition of *Essential Node* can classify a group of useful nodes in the tree. This is more meaningful then classifying a tree node as external or internal. The *Essential Nodes* highlight the most interesting part of the generic Chinese PAT tree. This property is very useful in our Chinese document clustering process.

A brief summary of comparison between the PAT tree, the PAT tree in reduced design, and our generic Chinese PAT tree is shown in Table 3.13.

	The PAT tree	**The PAT tree for Chinese in deduced design**	**Our generic Chinese PAT tree**
Text Component	Document Level	Sentence Level	Sentence Level
Node Type	Heterogeneous 1. Internal Node 2. External Node	Heterogeneous 1. Internal Node 2. External Node	Uniform Structure 1. Tree Node
Basic Node Components	Internal Node: 1. Check bit 2. Child pointers External Node: 1. Link to sistring	Internal Node: 1. Check bit 2. Child pointers External Node: 1. Link to sistring 2. Frequency count	Tree Node: 1. Check bit 2. Frequency count 3. Link to sistring 4. Child pointers *(With revised attributes)*
Essential Length	No	No	YES!
Essential Node	No	No	YES!
Tree Complexity	O(*n*)	O(*n*)	O(*n*)
Advantages	1. Compact in structure 2. Linear Storage Complexity 3. Quick Searching Time	1. Has all the advantages of the PAT tree 2. Capable of storing Chinese documents	1. Has all the advantages of the PAT tree 2. Capable of storing Chinese documents 3. In uniform structure 4. Utilize the usage of tree node content 5. Has Essential Node to represent the Chinese phrases 6. Easy to maintain 7. Easy to expand in functionality
Drawbacks	1. Cannot handle Chinese document correctly 2. May need some adjustments to suit a particular application need	1. Always require an additional tree root artificially 2. Does not obey the PATRICA structure 3. Ambiguous structure contents 4. Difficult to expand in functionality	1. Needs more memory for a tree nodes

Table 3.13 Summary of PAT tree and its Chinese variations

Chapter 4

Performance Analysis on the Generic Chinese PAT Tree

This chapter focuses on the performance and the usability of the generic Chinese PAT tree. We address the practical issues of the generic Chinese PAT tree, which include the run time analysis and the implementation analysis.

We conduct several experiments to evaluate the performance of the generic Chinese PAT tree. In our experiment setup, we prepare a set of articles collected from the local news sections in MingPao News [35]. The whole set contains 1750 pieces of news from 1st September 2001 to 30th September 2001. Each of the articles contains around 300 Chinese characters on average.

The experiments are performed on a 900MHz Pentium III with 256MB RAM running the Windows 2000 Operating System. Since Windows 2000 is a multitasking environment, we always conduct the same set of experiment on the same machine in the same period of time to minimize the randomness factor of the load on background jobs.

4.1 The Construction of the Generic Chinese PAT Tree

This experiment evaluates the actual running performance of our generic Chinese PAT tree. We partition the PAT tree construction process into three phrases. The first phrase is the loading of documents, the second phrase is the construction of the PAT tree nodes, and the last phrase is the detection of Essential Nodes. To simulate some operations acting on the nodes directly, we include a benchmark for quick sort on the essential nodes.

We expect that the main task in the PAT tree construction is to build and index the PAT tree nodes. Our result agrees with the expectation. From the result, the building of tree nodes consumes most of the computation time in the PAT tree construction process. It consumes about 70% of the construction time. The loading of documents into the memory, due to the technology advancement of the secondary storage device, is not a major factor affecting the runtime performance. It is increasing slowly, steadily, and linearly. The essential node detection takes some processing time, however, but it just plays in a minor role. When we construct the generic Chinese PAT tree for all the 1750 articles in the test set, the documents take about one second to load, the tree take about 12 seconds to build, and then the Essential Node detection takes about three seconds to complete. Figure 4.1 presents this experiment result.

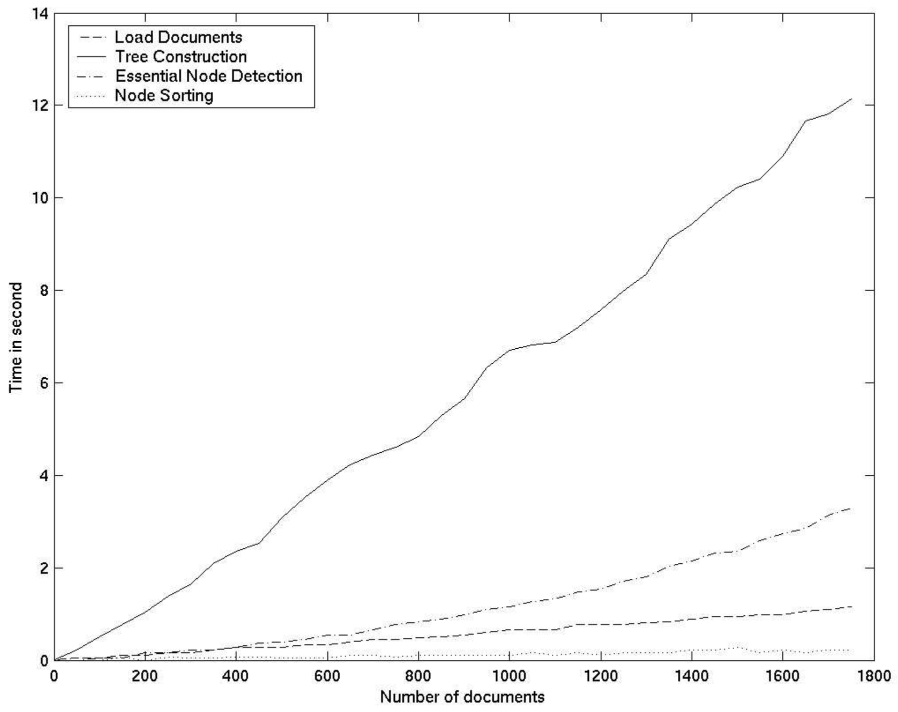


Figure 4.1 Run time for the generic Chinese PAT tree construction

In Figure 4.1, the black line shows the time for the generic Chinese PAT tree construction. Due to the non-uniform factor of the size of each document and the nature of the operating system in multi-tasking environment, the result we obtained is not steady as we expected. However, we still observe a linearly growing line. The building of PAT tree dominates the construction process. The run-time of other phrases is much lower than that of the tree building, and the Essential Node detection comes the second regarding the actual run-time. Although the process should be expected to perform in linear time, the result here does not produce sharp lines, due to the randomness of the document size and the tree shape. However, the growth of this curve is very slow and it will not affect the process time much.

To simulate some processing work in this experiment, we perform the sorting on the newly constructed PAT tree. We discover that the processing time is very fast. It is kept under one second through this experiment. This result reveals that operations on the generic Chinese PAT tree are usually fast, and the run time overhead is mainly in the building of tree nodes.

4.2 Counting the Essential Nodes

We conduct an experiment to illustrate the relation between the number of Essential Nodes and the number of leaf nodes in the generic Chinese PAT tree. In Figure 4.2, the dash line is a reference line to indicate the number of tree nodes in the generic Chinese PAT tree. The dotted line shows the number of leaf nodes, which is linearly related to the number of Chinese characters. These two lines represent the upper bound and the lower bound. The number of Essential Nodes should fall within this range. The figure shows that the number of Essential Nodes, surprisingly, is much closer to the number of leaf nodes, the lower bound side. Although the number of Essential Nodes is an arbitrary value that depends on the document context, our result shows that, in general, the number does not increase dramatically. This result provides a positive argument for the applicability of Essential Nodes in actual environment.

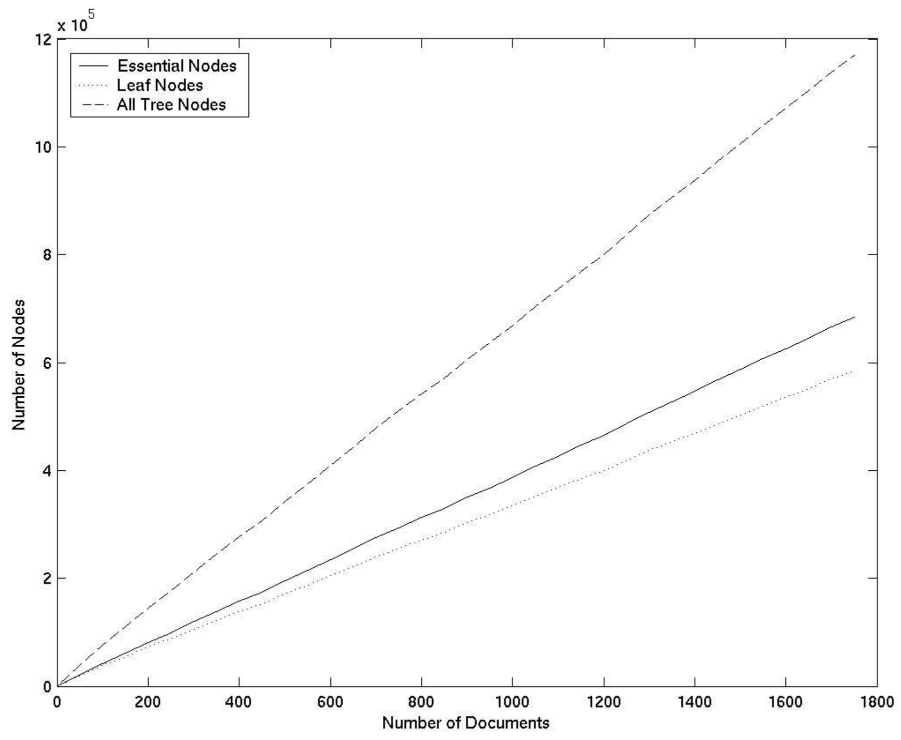


Figure 4.2 Relationship between number of Essential Nodes and leaf nodes

4.3 Performance of Various PAT Trees

To compare the run-time performance of the various PAT trees, we perform an experiment to construct the PAT tree, the Chinese PAT tree, and the generic Chinese PAT tree for the same set of articles. The experiment compares the building time of tree node since it is a necessary operation for all kinds of PAT trees. The building process is also a major factor affecting the run-time performance. Because of the extended structure of the generic Chinese PAT tree, we expect the building time, in exchange with new functions and abilities, to be a bit longer than the Chinese PAT tree. On the other hand, the original PAT tree performs worse than any of the Chinese PAT trees because of the incompatibility with our Chinese documents in the test set. Since the original PAT tree does not care the double-byte nature of Chinese characters, it consumes extra time in generating unwanted sistring information for the PAT tree.

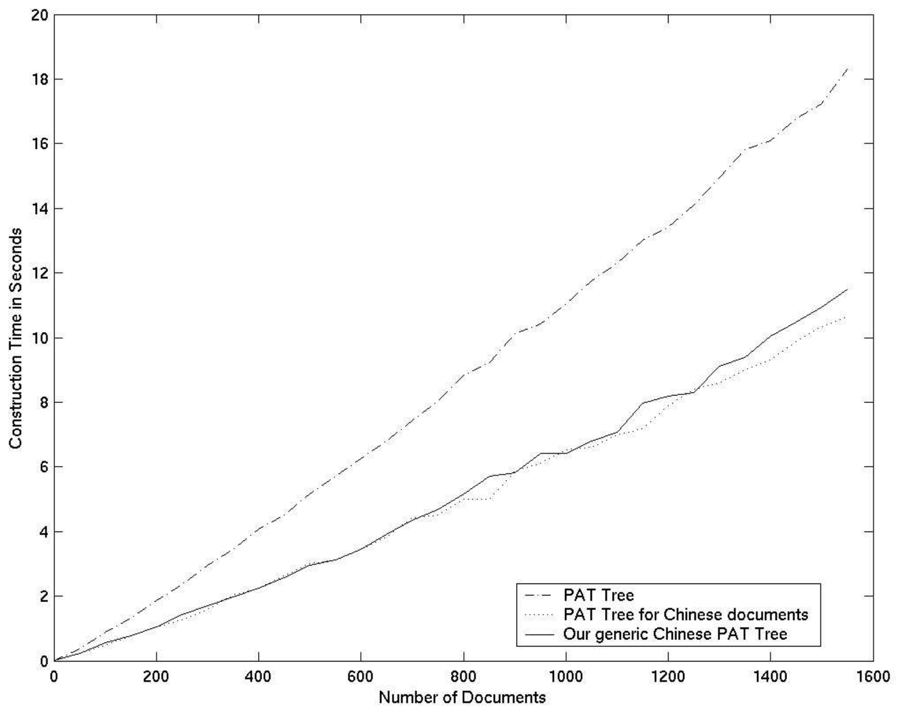


Figure 4.3 Construction time performance of various PAT trees

Figure 4.3 shows the tree building performance result. The PAT tree is the worst among them. Much of its time is wasted in constructing node for non-Chinese sistrings. Since the tree includes non-Chinese sistrings, we include this result mainly for comparison purpose. The resulting tree is, in fact, not useful for Chinese processing. For the Chinese PAT tree and the generic Chinese PAT tree, they are growing in the same rate while the generic Chinese PAT tree requires only slightly more building time compared with the Chinese PAT tree.

This result demonstrates that our generic Chinese PAT tree is a feasible structure. Although the run-time performance is affected by the increase in tree node size, which is due to the unification of node structure, the actual run-time does not increase too much when compared to the Chinese PAT tree.

4.4 The Implementation Analysis

In the previous section, we address the overhead issue on our new structure and conclude that the overhead is a controllable factor. However, we reveal an uncontrollable factor that affects all kinds of PAT tree during the experiments. That uncontrollable factor is due to the PAT tree implementation issue. Our first implementation attempt in dynamic memory allocation approach produces a very large clean-up overhead. Later we revise our implementation and propose the node production factory approach. We can then finally overcome this problem.

4.4.1 Pure Dynamic Memory Allocation

The common way to implement a tree structure is to use pointers. It is the most flexible and economical way of implementation. We allocate memory for each node during its insertion. The memory is released when we no longer require it in the tree, most likely at the end of the process. At the first glance, this dynamic pointer memory allocation approach sounds good and reasonable. It is also commonly used and widely adopted in handling list, stack and tree kind of data structures. However, in our experiment, we discover an unreasonable overhead of PAT trees in this implementation.

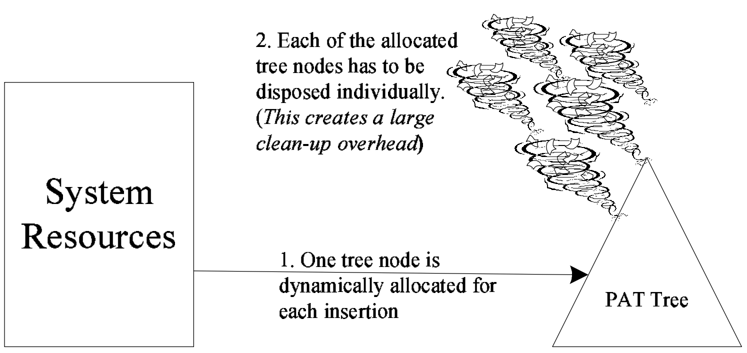

System Resources
1. One tree node is dynamically allocated for each insertion
2. Each of the allocated tree nodes has to be disposed individually. (*This creates a large clean-up overhead*)
PAT Tree

Figure 4.4 Dynamic memory allocation approach of PAT tree construction

Figure 4.4 illustrates the situation of the dynamic memory allocation approach. The tree node is allocated from the system resources on demand, one at a time, to build the PAT tree. After the process, the tree node has to be disposed, again, one by one. Thus the clean-up of tree node generates a very large run-time overhead. Figure 4.5 shows the overhead the dynamic memory allocation method introduces. The figure is similar to those in the previous section but we include the clean-up time. It shows that the clean-up time is far more than the actual processing time and even reaches an unacceptable value. While the tree node construction time is in the scale of several seconds, the clean-up time for a thousand documents can be as high as hundreds of seconds. Our result records up to one thousand documents only, because the result beyond that takes even more time. From this observation, we discover that the system takes a long time to clean-up the dynamically allocated memory. When we declare hundreds or thousands of tree nodes, the process takes a huge amount of time to release that hundreds or thousands of memory allocations piece one by one.

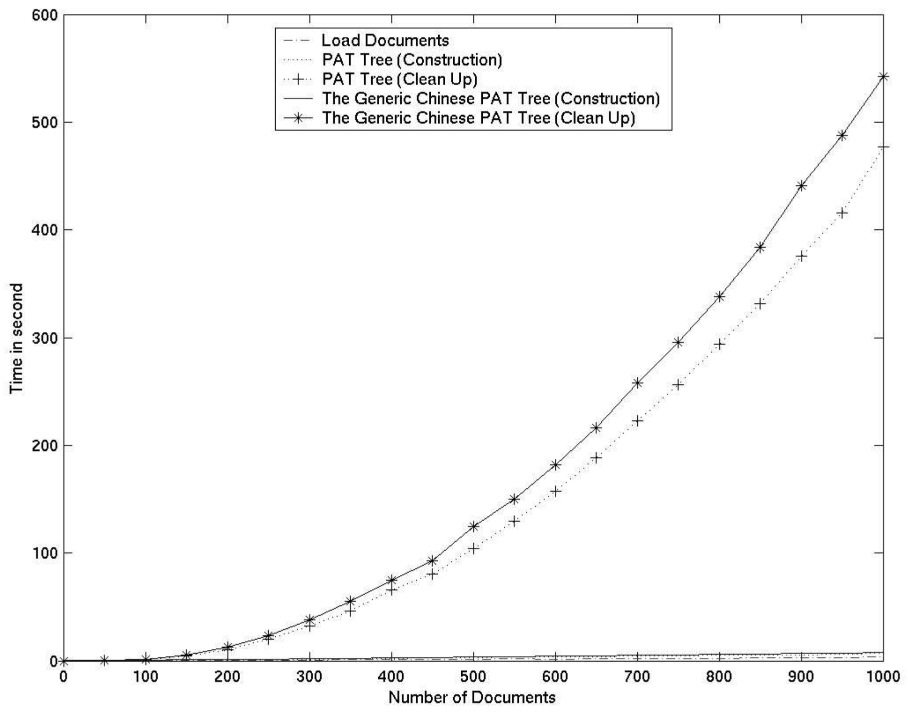


Figure 4.5 Clean-up overhead of dynamic memory allocation approach

4.4.2 Node Production Factory Approach

To overcome the tedious clean-up overhead problem, we introduce a factory approach to regulate the PAT tree construction process. Our concept is to create a node production factory behind the PAT tree. The node production factory, which acts as the primary source of tree nodes, reserves a larger piece of memory to produce a number of tree nodes. It is a one-time reservation of memory so that the system needs to handle the clean-up of memory one time only, instead of hundreds or thousands of times. In the new implementation, every time we need to create a tree node, instead of allocating it dynamically, we ask the node production factory to produce a pre-manufactured tree node. The tree should return the tree node to the node production factory after its use. In short, the tree structure and the involved tree algorithm do not change, but the memory allocation is done via the help of the node production factory. Figure 4.6 illustrates the concept of node production factory.

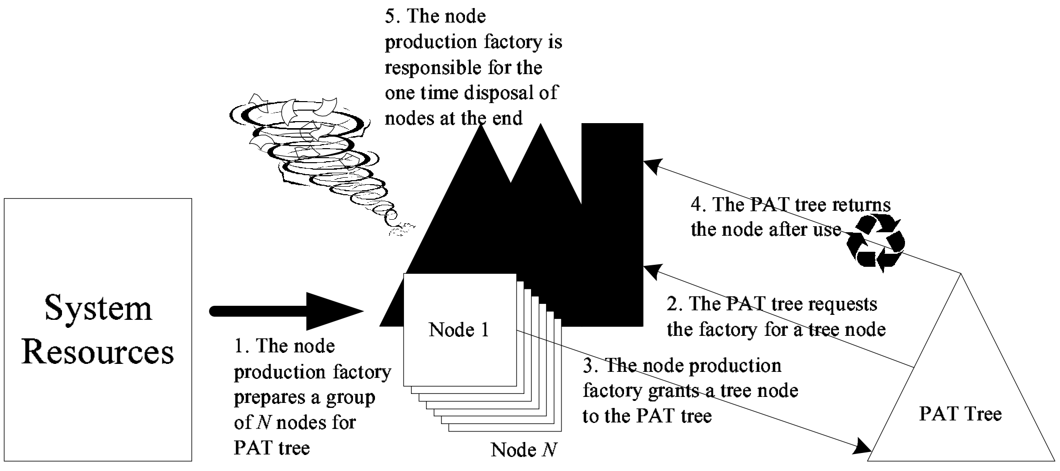

5. The node production factory is responsible for the one time disposal of nodes at the end
System Resources
1. The node production factory prepares a group of *N* nodes for PAT tree
Node 1
Node *N*
4. The PAT tree returns the node after use
2. The PAT tree requests the factory for a tree node
3. The node production factory grants a tree node to the PAT tree
PAT Tree

Figure 4.6 Node production factory approach of PAT tree construction

The idea of node production factory is a trade-off between the static memory allocation approach and the dynamic memory allocation approach. In the pure static memory allocation approach, we need to reserve a large enough memory for our need. This limitation makes it unsuitable for PAT tree. On the other hand, the pure dynamic memory allocation approach gives an unreasonable overhead during run time. The result makes the pure dynamic memory approach impractical. Node production factory is a solution to overcome the problems. We can define a size parameter (*N*) for the node production factory to indicate the number of tree node it will produce. When *N* is large, it is more capable of handling a larger document, but the memory is wasted when the document is actually small.

Although the node production factory is a remedy to the clean-up overhead problem, we have no way to determine a suitable *N* for one factory that fulfills all possible situation at all time. In order to make the node production approach adaptive, we can include a chain of factories to backup the main factory. When the nodes in the main factory are used up, it breeds a new identical factory to supply a new stock of tree nodes. This approach is recursive such that each factory, with the help of its associated sub-factory, can theoretically supply any number of tree nodes, up to the physical size of memory. When *size=1*, the factory chain would be a special situation that is similar to the pure dynamic memory allocation approach. Figure 4.7 shows the idea of a chain of factory.

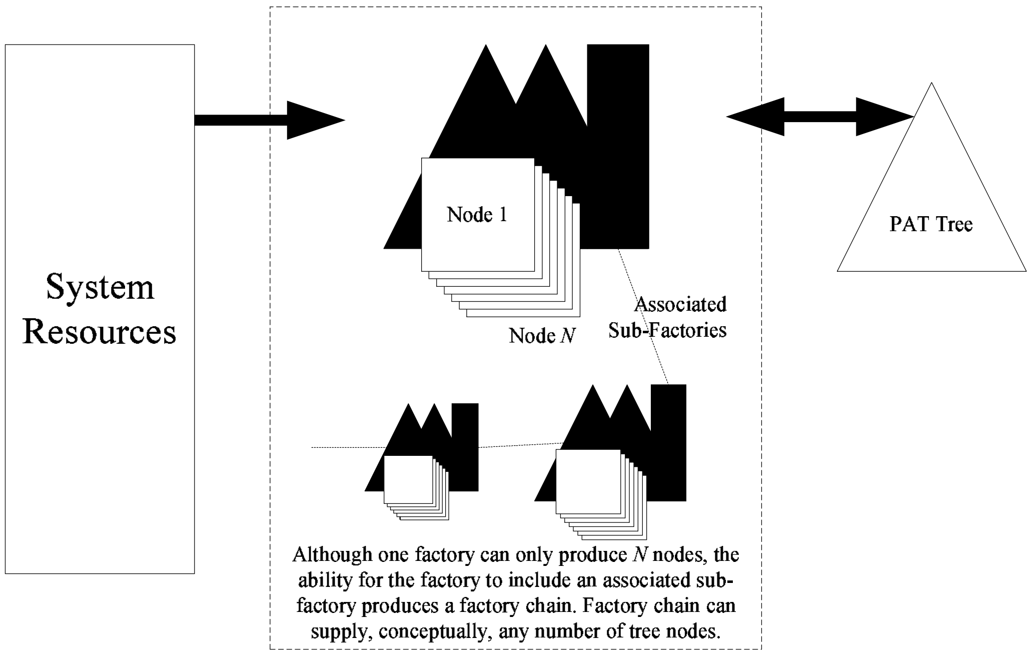

System Resources
Node 1
Node N
Associated Sub-Factories
PAT Tree
Although one factory can only produce N nodes, the ability for the factory to include an associated sub-factory produces a factory chain. Factory chain can supply, conceptually, any number of tree nodes.

Figure 4.7 The factory chain concept of node production factory approach

4.4.3 Experiment Result of the Factory Approach

We conduct an experiment to show the effect of the node factory implementation. The result is presented in Figure 4.8. We can compare Figure 4.8 with Figure 4.5 to show the improvement in using the node production factory approach. The construction time in the node production factory approach is shifted by a constant time about one second, due to the initialization of the factory. However, it pays off soon because the clean-up time is kept at a very low constant value of about 0.33 seconds. Compared with the dynamic memory allocation implementation, this is a significant run-time improvement. PAT tree becomes practically usable in this approach. By tuning the size of production parameter (*N*), we may reduce the overhead on the factory initialization. A factory with a large *N* produces more tree nodes that require less spawning of associated sub-factories, and a factory with smaller *N* produces less tree nodes that may improve the overhead on the factory initialization.

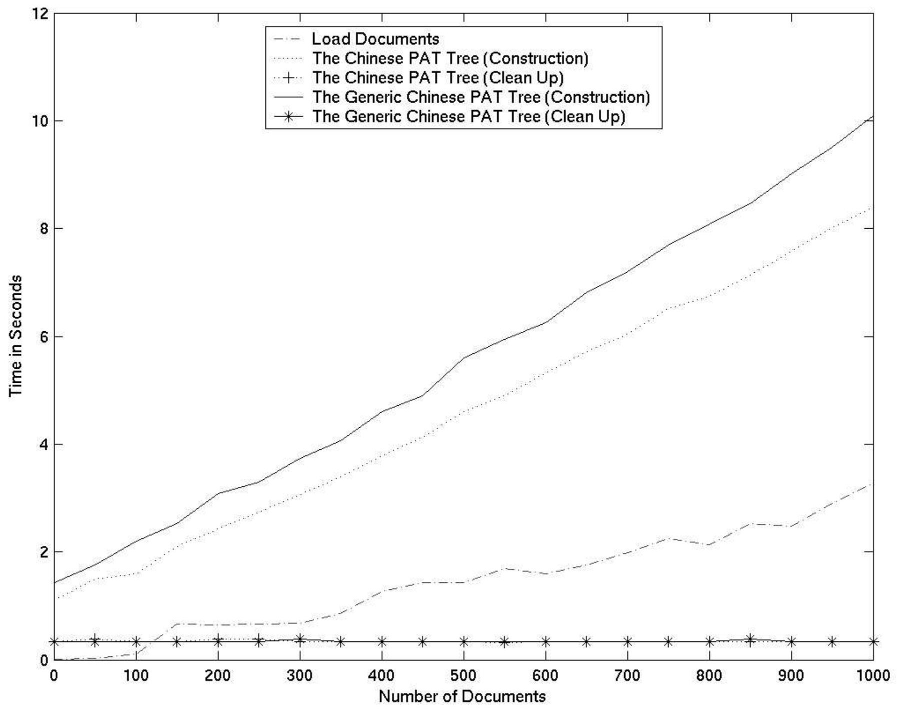


Figure 4.8 Clean-up time of node production factory approach

This result shows the importance of implementation design of the PAT tree. A general tree implementation approach using dynamic memory allocation of node on demand will significantly affect the overall processing time for real-life applications. The node production factory implementation is a trade-off solution. Since we must maintain and provide an additional tree node factory to control the usage of node, our solution requires some more tree construction time. However, this overhead is not significant when we compare it with the clean-up time overhead in the dynamic memory allocation approach.

Chapter 5

The Chinese Documents Clustering

In this chapter, we discuss the design of our clustering framework that can classify a set of Chinese documents into clusters with interesting topics. We describe the overview and details in our clustering framework, and give an overall picture in our clustering process.

5.1 The Clustering Framework

We propose our method for the Chinese document clustering. Document clustering is a problem to separate a collection of documents into groups such that each of the groups forms a cluster. Documents inside a cluster should contain similar content or point of interest. The clusters may not be partitions of the document collection. A document in the collection set is possible to be categorized into several clusters. For example, a collection of articles from sport magazine may be categorized into different kind of sports like football, basketball, and more. It can also be categorized into events that happened in each year. However, if it is a sport magazine related to NBA basketball, it is obviously meaningless to categorize the articles into football, basketball, volleyball, and others since all articles is generally basketball news. On the other hand, if the articles do not come with the chronological information, none of the articles can be in the category of any year. This example illustrates the situation that a predefined category for the group separation may not be appropriate.

Our clustering involves the following steps:

1. Documents cleaning,
2. PAT tree construction,
3. Essential Node extraction
4. Base clusters detection
5. Base clusters filtering
6. Base clusters grouping
7. Documents assigning
8. Result presentation

The first three steps are the data preparation process on the raw collections of Chinese document. Step four to step six are the process to identify the cluster bases. Step seven is the clustering process to group documents together according to the cluster bases. The final step deals with the formatting issues for review and visualization. Figure 5.1 gives an overview of the clustering flow.

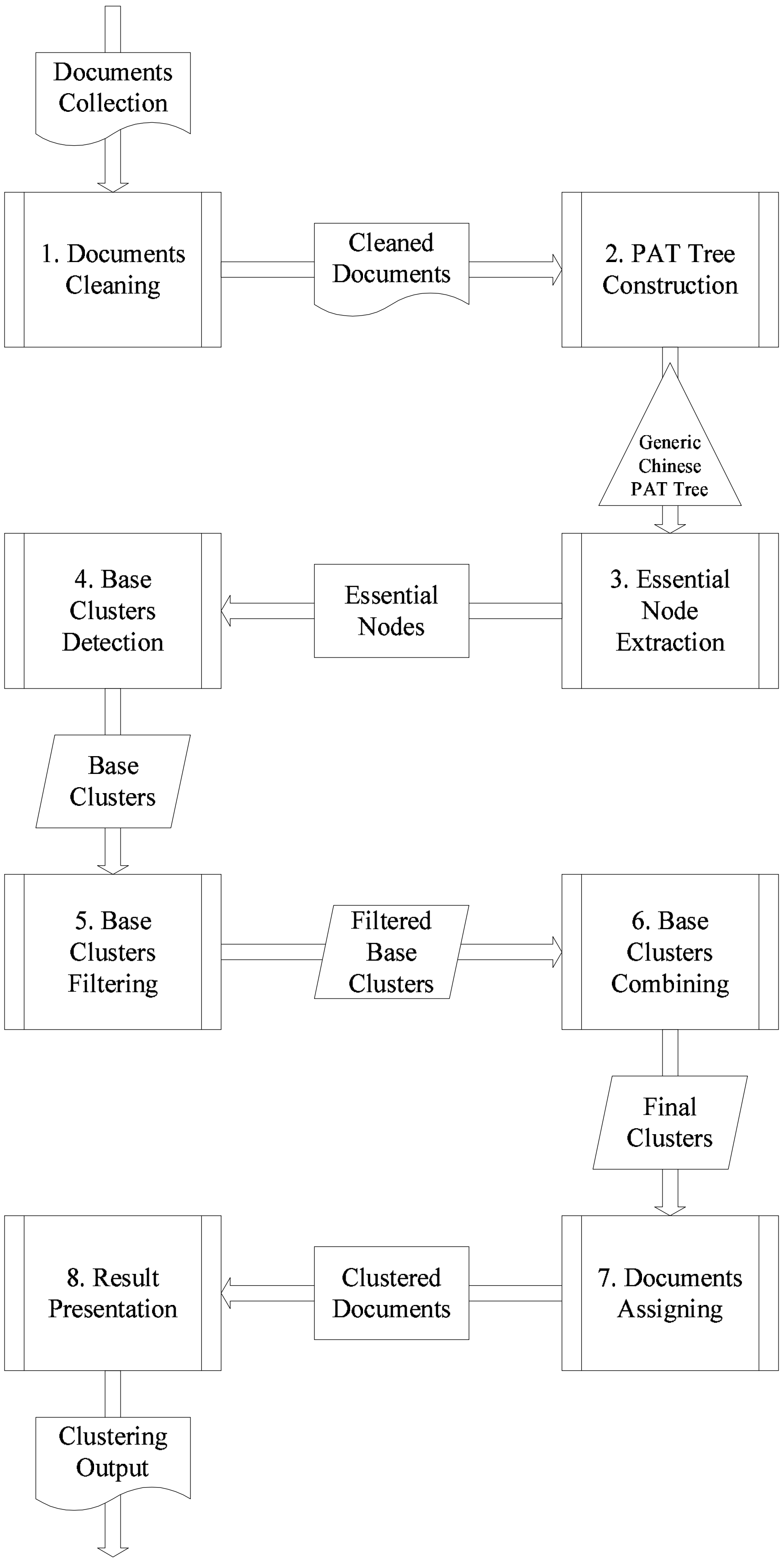


Figure 5.1 Flow diagram of the Chinese documents clustering process

5.1.1 Documents Cleaning

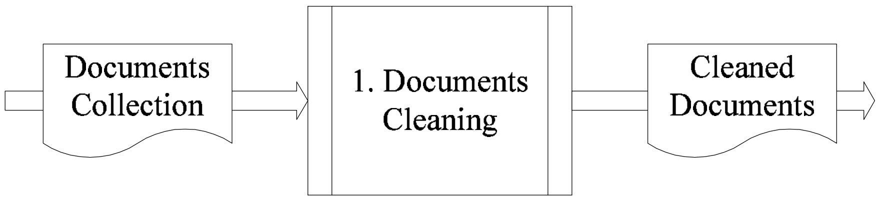


Figure 5.2 Documents cleaning process

Figure 5.2 shows the documents cleaning process. The documents collection for the clustering can be in various formats. They may contain non-Chinese characters as well as some unwanted information, such as the signature label, time stamp, or copyright notice, etc. The cleaning of data is an important pre-processing step to take out the unwanted information from the raw data. Since the formats of the documents collection can be different, there is no universal cleaning filter for the clustering process. Sometimes we require several filters to complete the cleaning process. We discuss the cleaning of dedicated Chinese documents and Chinese Web documents as examples of the cleaning process.

5.1.1.1 Cleaning of Dedicated Documents

Dedicated documents are Chinese documents containing pure Chinese contents. We assume that the documents do not contain any non-Chinese contents. In other words, the documents should be error-free. In the cleaning process, we apply the segmentation filter to remove all the punctuation marks. According to the punctuation symbols, we decompose the document into chunks of Chinese sentences.

Figure 5.3 illustrates the basic function of the cleaning process. In this example, we have a document in the collection with a simple sentence. The cleaning process applies the segmentation filter. It treats the punctuation marks as delimiter, and three short sentences are detected. The output is three chunks of Chinese sentences. Each of them is punctuation free and ready for the next step.

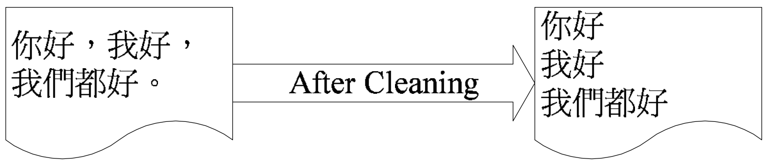


Figure 5.3 Example of a dedicated document after cleaning

5.1.1.2 Cleaning of Web Documents

Although the previous data cleaning process is very simple, we are unlikely to obtain the error-free document in the real situation. When documents are collected from the Web, it is more likely that contents may contain noises and errors. In this situation, the cleaning process becomes more important. As we cannot guarantee the documents to be error-free, we need some noise removal and error detection techniques during the data cleaning process. For example, English and Chinese characters may exist and interleave each other. One byte of the double-byte Chinese character may be missing due to various unknown errors from the source. When we tried to collect documents from various Web sites, including Apple Daily Online [3], MingPao News [35], Oriental Daily News [39], and The Sun Web [46], we found that their content structures are different. Therefore, we need to customize the cleaning process for documents from a particular site. There is no single universal filter to clean up them all. However, we still can come up with a generic flow to filter the web documents.

Web documents are in HTML files. When we need to clean up the web documents, we need to apply several filters on the cleaning process. They are:

1. Site-Content Filter,
2. HTML Filter,
3. Error Correction Filter, and
4. Segmentation Filter.

We first apply the site-content filter to remove the decorative details of the document, including, but not limited to, advertising banners, navigation bars, images, and image descriptions. All these decorations are part of the HTML file but not part of the content. This filter is dedicated to documents from one type of source but not reusable for documents from another source.

The result from the site-content filter is an HTML document containing only the text content. We apply the HTML filter to the result to obtain the text document. The HTML filter can remove all the unnecessary HTML tags from the HTML document. It can also correctly convert the HTML newline symbol (`<br>`) to line break in the output text document. As a result, the output from the HTML filter is a plain text of Chinese document.

Since the web document may contain unpredictable character and the filtering process may also inject some errors, we apply an error correction filter to filter out any suspicious symbols from the plain text document. The filter will ensure the content to be limited to Chinese characters and the punctuation mark symbols that we allow it to retain. As a result, all unrecognized characters are removed from the text, including those Chinese words that miss a single byte of content.

Finally, we apply the segmentation filter as previously mentioned to break down the text into document level. The obtained result after cleaning is ready to process in the next step.

5.1.2 PAT Tree Construction

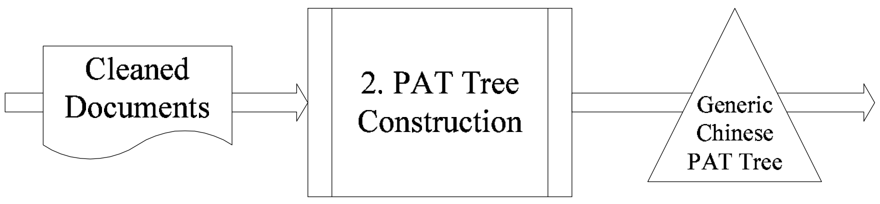


Figure 5.4 PAT tree construction process

In the PAT tree construction process, shown in Figure 5.4, we insert the sistrings of the cleaned documents to the generic Chinese PAT tree. The insertion of sistrings is a fast operation in $O(\log m)$ where m is the size of the PAT tree. The PAT tree provides information to guide the clustering process. It provides various information about the documents collection. Each node in the generic Chinese PAT tree contains a check bit, a link to sistring, a frequency count, and a pair of pointers to its left child and right child. Besides these basic components, we also record the number of documents (*N*) in each node, indicating the number of documents, which contain the sub-strings represented by that node. For example, if there are three documents, the first one includes '大學生' once, the second one includes '大學生' two times, and the third one includes no '大學生' in the text, then we have a tree node in the generic Chinese PAT tree representing '大學生'. The number of documents (*N*) in that tree node is two, indicating that there are two documents in the collection contain '大學生'.

We also need some other measure of collection information for the clustering need. These measurements are easy to obtain from the generic Chinese PAT tree. When we discuss the possible clustering functions, we give details of these measurements and we show the way to obtain them from the generic Chinese PAT tree.

After we construct the generic Chinese PAT tree successfully, we can then feed the resultant tree to the next step for clustering.

5.1.3 Essential Node Extraction

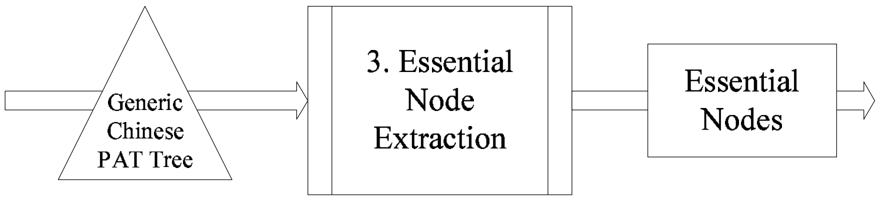


Figure 5.5 Essential Node extraction process

Essential Node Extraction, as shown in Figure 5.5, is a process to detect Essential Nodes from the generic Chinese PAT tree. Since each essential node can uniquely identify a Chinese phrase in the document collection, the extraction process can effectively retrieve all Chinese phrases related in the Essential Nodes.

The Chinese phrases contained in the Essential Node must be syntactically correct, for example, '計算機' may be a phrase in an Essential Node *A* while '算機' may also be a phrase in an Essential Node *B*. If there is another node *C* that can represent '計算機' only, it will duplicate the node *A* and it is not an Essential Node (otherwise node *A* is not an Essential Node). If there is a node *D* that can only represent the bit pattern, say, '`10011110 1000`', it cannot represent any Chinese phrase, so it must not be an Essential Node.

The Essential Node Extraction is a very important step in the clustering process. This process can extract the most useful information, which is the Chinese phrases, from the document collections. When we consider the Essential Node, the result is with respect of the Chinese information inside the tree. On the other hand, if we consider the content of the whole PAT tree for the clustering, those unrelated nodes might contribute to the result. The unrelated nodes that do not contain Chinese information will affect the result.

In order to capture the Essential Nodes from the generic Chinese PAT tree, we compare all the tree nodes against their ancestor node by the rules defined in Section 3.3.3.2. A node (*x*) is an *Essential Node* (*EN*) if and only if its

(1) $\textit{Essential Length}_x \geq 32$; and

(2) $\textit{Essential Length}_x - \textit{Essential Length}_y \geq 16$ (if such y exists),

where node (y) is the nearest ancestor of x such that y is also an *EN*.

In the Essential Node detection process, we initially assign all each tree node as a non-Essential Node. We traverse the generic Chinese PAT tree from the root node in a pre-order sequence. At the beginning, we apply the first rule to check against every tree node. When the node contains an Essential Length greater than 32, we mark it as an Essential Node. For the nodes under this tree node, we use the second rule to determine whether the node is an Essential Node. Since the detection process is deterministic upon pre-order traversal, we only need to seek every node once to determine the Essential Nodes from all tree nodes. The algorithm for the Essential Node detection is shown in Figure 5.6.

```
Essential_Node_Detection()
    Essential_Node_Detection_Rule_1(tree.Root)
End

Essential_Node_Detection_Rule_1(x)
    x.essential ← FALSE
    if (x.essentialLength >= 32)
        x.essential ← TRUE
        Essential_Node_Detection_Rule_2(x.path0, x.essentialLength)
        Essential_Node_Detection_Rule_2(x.path1, x.essentialLength)
    otherwise
        Essential_Node_Detection_Rule_1(x.path0)
        Essential_Node_Detection_Rule_1(x.path1)
    endif
End

Essential_Node_Detection_Rule_2(x, y_length)
    x.essential ← FALSE
    if (x.essentialLength - y_length >= 16)
        x.essential ← TRUE
        Essential_Node_Detection_Rule_2(x.path0, x.essentialLength)
        Essential_Node_Detection_Rule_2(x.path1, x.essentialLength)
    otherwise
        Essential_Node_Detection_Rule_2(x.path0, y_length)
        Essential_Node_Detection_Rule_2(x.path1, y_length)
    endif
End
```

Figure 5.6 Essential Node detection algorithm

5.1.4 Base Clusters Detection

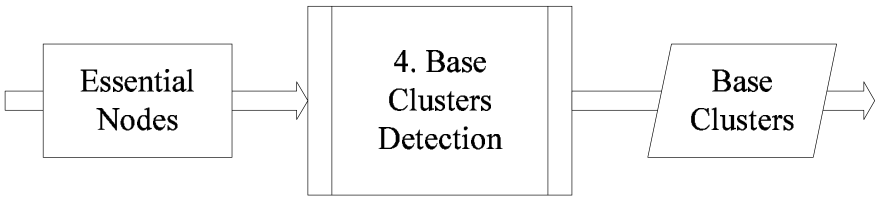


Figure 5.7 Base clusters detection process

Base Clusters Detection, a process shown in Figure 5.7, is the most important step in our clustering process. This process attempts to detect clusters from the document collection. It analyzes the Essential Nodes among the generic Chinese PAT tree. The base clusters are a subset of the Essential Nodes and the clustering algorithm determines the base clusters result.

To determine the base clusters from the Essential Nodes, the process basically involves the computation of a weight value. The importance of each Essential Node depends on its weight value. The higher the weight value, the greater the chances that Essential Node contains a representative phrase as a base cluster.

We study several ranking functions to compute the weight of the Essential Nodes. While there is no objective measurement for the goodness of each ranking function, we analyze the characteristics of these functions to distinguish their strengths and weaknesses.

5.1.4.1 Frequency Count

Frequency count is one of the basic components in the generic Chinese PAT tree. It indicates the occurrences of a node in the tree. In other words, it reveals the popularity of a phrase represented by the Essential Node. Therefore, frequency count is one of the simplest metrics to determine the weight value. For each phrase *p* in a document *i*, we denote its frequency as

$$frequency_p^i$$

when we consider the whole document collection, we denote the frequency of a phrase *p* in the document collection as

$$frequency_p = \sum_{i=1}^{N}(frequency_p^i)$$

Since the above frequency is equivalent to the frequency count of the Essential Node representing the phrase *p*, the weight value using the frequency as ranking function would be

$$weight_p = frequency_p$$

As the frequency reflects the popularity of phrases, this ranking function can efficiently remove those insignificant phrases from the popular ones. Besides, it involves minimal computation because the frequency count is the default value existing in each node of the generic Chinese PAT tree. However, using the frequency count directly as the ranking criteria contains several disadvantages. It is sensitive to several kinds of noise. If the document collection is noisy with several documents containing unusually high occurrences of some phrases, their frequency counts will be increased drastically to affect the ranking result. Besides, the ranking result is in favor of long documents: the phrases in a long document tend to have higher frequency. In addition, phrases with a very high frequency in a single document may rank high in this ranking function, but such a phrase is obviously not suitable to become a base cluster. Therefore, although direct measurement of frequency count is a simple ranking method in favor of computation time, it is too sensitive to various kinds of noise. The advantages and disadvantages are listed in the Table 5.1.

Advantages	✓ Frequency count is an obvious measurement ✓ No additional calculation is needed
Disadvantages	✗ Sensitive to noise in a document ✗ Sensitive to irrelevant documents ✗ Sensitive to the size of document ✗ Long document may have phrases that dominate the ranking

Table 5.1 Summary of the frequency count ranking method

5.1.4.2 Total Term Frequency (ttf)

Total term frequency is another measurement for the ranking of base clusters. The term frequency is a value to indicate the importance of a phrase in a particular document. By definition, the term frequency of a phrase *p* in the document *i* would be,

$$tf_p^i = \frac{frequency_p^i}{\max\limits_p(frequency_p^i)}$$

The nominator is the frequency count of the phrase *p* in the document *i,* and the denominator is the maximum of these frequency counts in the document *i*. If the phrase *p* does not exist in the document *i*, its term frequency tf_p^i would be equal zero. If the phrase *p* is the most dominating phrase in the document *i*, its term frequency tf_p^i will be equal to one.

In order to determine the importance of a phrase *p* in the documents collection, we define the total term frequency by summing up the term frequencies in all documents together,

$$weight_p = ttf_p = \sum_{i=1}^{N}(tf_p^i)$$

The *ttf* is a better ranking criterion over the simple use of frequency count. It is because the value of term frequency is a polished value of the raw frequency count. It can avoid an over-emphasis of frequency count in a long document. Besides, this value can reflect the importance of one phrase to another in a single document, so that it prevents long documents from dominating the ranking result of base clusters. However, term frequency is not a measurement directly available from the generic Chinese PAT tree. We need to construct an additional generic Chinese PAT tree for each document in order to collect the term frequency in each of them. Besides, there are still many disadvantages in using total term frequency as in the frequency count method. The summary of advantages and disadvantages of ranking by total term frequency is listed in Table 5.2.

Advantages	✓ It is a reasonable ranking method ✓ It can avoid the result dominated by long documents
Disadvantages	✗ Still sensitive to noise in a document ✗ Still sensitive to irrelevant documents ✗ To measure the value, we require an additional generic Chinese PAT tree for each individual document

Table 5.2 Summery of the total term frequency ranking method

5.1.4.3 Inverse Document Frequency (idf)

Inverse Document Frequency is a measurement of how rare the phrase occurs in the collection of document. The value of inverse document frequency of phrase *p* is defined by

$$idf_p = \log(\frac{N}{n_p})$$

where N is the total number of documents in the document collection and n_p is the number of documents in the document collection that contains the phrase p.

If a phrase p appears in every document, N and n_p are equal and idf_p becomes zero. This implies p appears too frequent so p is not significant to be a base cluster. Therefore, the inverse document frequency is a common modifier applies together with the total term frequency. The ranking function then becomes,

$$weight_p = ttf_p \times idf_p$$

The ranking function that involves the inverse document frequency is commonly used in document clustering. The definition of ttf_p and idf_p, however, is contradicting. When a phrase p has high ttf_p, it is more likely to have high n_p, resulting in a very low idf_p. On the other hand, a phrase p with low ttf_p is more likely to have low n_p, so the idf_p is usually high. When either ttf_p is very low or idf_p is very low, $weight_p$ would be low. Therefore this ranking function is not in favor of either extreme.

ttf_p	idf_p	$weight_p$	**The possible nature of the phrase** p
LOW	LOW	LOW	Insignificant to the documents collection May not have rich meaning semantically
LOW	HIGH	LOW	Insignificant to the documents collection May have rich meaning semantically
HIGH	LOW	LOW	Common in natural language May not have rich meaning semantically (e.g. 我們)
HIGH	HIGH	HIGH	Common in the documents collection Should have rich meaning semantically *(So it is suitable to be base cluster)*

Table 5.3 Interpretation of ttf_p and idf_p in the ranking function

Although this ranking method is reasonable and it is popular in clustering problems, the method is not quite suitable for our problem. The value of ttf_p and

idf_p are not in the same scale. While the ttf_p linearly reflects the number of occurrences with no upper bound, idf_p is a logarithmic value always bounded by $\log(N)$ where N is the number of documents in the collection. Therefore, the importance of ttf_p and idf_p is not equal but depends on the size of the collection. ttf_p plays a very important role in a small document collection and the effect of idf_p slowly increases when the size of the collection increases.

This ranking method is described in [29]. From the numerous testing result with this method, we discover this unbalanced nature for ttf_p and idf_p. We attempt to normalize the idf_p in the range between 0 and 1 inclusively to avoid its uncertain upper bound. However, this modifier reduces the influence of idf_p in the weight function. We then attempt to normalize ttf_p into a bounded range $0 \le ttf_p \le 1$ by dividing its value by $\max(ttf_p)$. However, the normalization involves heavy computation and the obtained result does not show much improvement.

5.1.4.4 Weighted Frequency

Weighted frequency is an adjusted value from the frequency count. We define it as

$$weight_p = frequency_p \times (baseScore_p)^2$$

$$\text{where } baseScore_p = \max(ChineseCharacterLength_p, topScore)$$

The $baseScore_p$ is an adjustment value of the $frequency_p$. It is based on the length of the phrase, in terms of the number of Chinese characters. The reason for this adjustment is because a longer phrase tends to contain richer meaning and is more likely to be a base cluster. At the same time, we know that a meaningful and representative phrase should not be a very long phrase [56], so the $baseScore_p$ limits the upper bound of $baseScore_p$. In Chinese documents, a vocabulary usually consists of two to three Chinese characters (e.g. 天氣, 旅遊,

交通), and names and special phrases usually consist of three to four Chinese characters (e.g. 董健華, 陳方安生, 桃李滿門). Sometimes a meaningful short phrase contains six or more Chinese characters (e.g. 香港中文大學, 書中自有黃金屋).

Frequency is one of the most relevant measurements to indicate how important a phrase is [17]. But since the $baseScore_p$, which is related to the phrase length, is additional information related to representative power of a phrase, weighted frequency is a better ranking function over the total term frequency. On the other hand, since the computation of weighted frequency is easy and straightforward, it is a more preferable ranking function than the *ttf-idf* function.

Although the weighted frequency has its advantages over the other ranking measurement, it still suffers from the noise problems. Some frequently appearing word combinations may not be any meaningful words but are still ranked high. For example, '們的'can be possibly in high ranking if there are quite a number of '我們的', '你們的'..., etc., and '們的' is not meaningful to be a cluster. To avoid this situation, we have to apply filtering techniques after this ranking process. In the next section we discuss some segmentation techniques that can help in the word filtering for the next process phrase.

5.1.5 Base Clusters Filtering

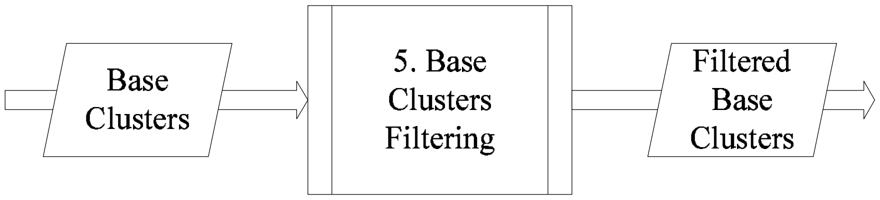


Figure 5.8 Base clusters filtering process

The base clusters from the previous process may not be noise-free, therefore, the base clusters filtering step is responsible for filtering out the improper base clusters that are wrongly detected in the previous step. This process is shown in Figure 5.8. In this filtering step, we apply the sentence segmentation techniques to verify if the phrase in the base clusters is a valid and complete phrase.

This filtering is based on the contextual information of the document collection. By inspecting the phrase itself without language knowledge, we cannot know if the phrase is a real Chinese phrase with actual meaning or if it is just a group of character with no meaning at all. However, with the contextual information, an important real phrase is more likely to appear together again and again. We study some algorithms that serve such purpose. These include the Significant Lexicon Patterns (SLP) analysis [14], Left Context Dependency/Right Context Dependency (LCD/RCD) analysis [17], and Complete Lexicon Patterns (CLP) analysis [16].

5.1.5.1 SLP Analysis

SLP analysis [14] focuses on the mutual information on the text. Its aim is to detect significant phrases, which are self-contained in certain level. The candidates are selected when they are not likely to be part of a longer phrase.

SLP analysis is a filtering algorithm based on the significance estimation (SE) function. When there is a phrase c composed of the lexicon pattern $\mathrm{c}_1, \mathrm{c}_2, ..., \mathrm{c}_\mathrm{n}$, we have two longest composed sub-strings of c with the length n-1. These two sub-strings a, b are in the form of $\mathrm{a} = \mathrm{c}_1, \mathrm{c}_2, ..., \mathrm{c}_{\mathrm{n-1}}$, and $\mathrm{b} = \mathrm{c}_2, \mathrm{c}_3, ..., \mathrm{c}_\mathrm{n}$. The definition of SE for the phrase c is

$$SE_c = \frac{f_c}{f_a + f_b - f_c}$$

where $f_a, f_b, \text{and } f_c$ are the frequencies of phrases $\mathrm{a, b, and\, c}$, respectively.

The algorithm of SLP analysis is outlined in Figure 5.9. The second and third rules of the algorithm make use of the SE function for the analysis.

```
SLP_Detection_Algorithm()
    foreach (node_i) in tree
        node_i.SLP ← UNKNOWN
    SLP_Detection_Rule_1(tree.Root)
End

SLP_Detection_Rule(a)
    if (a.essential is TRUE) and (a.frequency >= $frequency_{threshold}$)
        if (a.SLP is not FLASE)
            SLP_Detection_Rule_1(a)
        endif
    endif
    if (a is not leaf)
        SLP_Detection_Rule(a.path0)
        SLP_Detection_Rule(a.path1)
    endif
End

SLP_Detection_Rule_1(a)
    if (a is leaf)
        a.SLP ← TRUE
    else
        c ← one_of_the_successor_essential_node(a)
        b ← node_with_phrase($c_2,c_3,\ldots,c_n$)
        SLP_Detection_Rule_2(a,b,c)
    endif

End
```

```
SLP_Detection_Rule_2(a,b,c)
    $SE_c$ = c.frequency / (a.frequency + b.frequency – c.frequency)
    if ($SE_c$ >= $SE_{threshold}$)
        a.SLP ← FALSE
        b.SLP ← FALSE
    else
        SLP_Detection_Rule_3(a,b,c)
    endif
End

SLP_Detection_Rule_3(a,b,c)
    $SE_c$ = c.frequency / (a.frequency + b.frequency – c.frequency)
    if ($SE_c$ < $SE_{threshold}$)
        if (a.frequency >> b.frequency)
            a.SLP ← TRUE
        endif
    endif
End
```

Figure 5.9 SLP detection algorithm

This method helps eliminating some incomplete phrases inside a more complete phrase. However, this method cannot comprehensively consider all the situations for the phrases. For example, when we consider a phrase a = “糯米”, phrase c can possibly be “糯米飯”, or phrase c can possibly be “糯米糕” too. The algorithm contains no restriction on the selection of phrase c, and it considers only one possible phrase c. In this situation, the analysis results of choosing “糯米飯” or “糯米糕” or others can be greatly different. If SE_c for “糯米飯” is greater than $SE_{threshold}$ (rule 2), than the phrase a is “糯米” and the phrase b is “米飯”. Both phrase a and b will not be considered as SLP.

When we choose phrase c as “糯米糕”, the immediate difference we can notice is that the phrase b will not be “米飯” anymore. Since b depends on c, when c is “糯米糕”, b will be “米糕”. In this case, we evaluate the “糯米”, “米糕”, “糯米糕” combination and ignore the “糯米”, “米飯”, “糯米飯” combination. As a result, we fail to take out “米飯” as non-SLP. This leads to incomplete analysis of the phrase “米飯” eventually. In fact, when we choose phrase c as “糯米飯”, we also miss the evaluation of b as “米糕”. Therefore, this SLP algorithm cannot perform a complete analysis for every situation and the result can be arbitrary. It is possible that quite a number of phrases remain uncertain because of this problem.

We work on improving the SLP algorithm to make it more complete and accurate. We attempt to consider all the possible a, c pairs so that we will not miss any possible a, b, c combination for the analysis. However, this leads to another difficulty in resolving conflicts. Consider if the “糯米”, “米糕”, “糯米糕” combination with SE_c is less then $SE_{threshold}$ (rule 3), depending on a.frequency and b.frequency, phrase a “糯米” may be considered as SLP or remains uncertain at this time. It conflicts with the “糯米”, “糯米飯”, “米飯” combination that “糯米” is non-SLP. Since there is no order for the a, b, c combinations with the same phrase a, each result cannot override one another. Although we performed many case studies on the SLP with complete phrase c analysis, it still cannot provide a better solution to solve these conflicts.

Currently, the SLP analysis is useful in certain level but it contains limitations. As there is no significant improvement over the SLP algorithm, we require a better analysis technique to perform the base clusters filtering.

5.1.5.2 LCD/RCD Analysis

LCD/RCD analysis [17] is a measurement of external dependency of a phrase. The method bases on the belief that meaningful phrase would be self-contained. So we expect to see a rich variety of characters in the phrase boundary. In other words, when there are always the same few varieties of characters adjacent to a particular phrase, that phrase should be dependent on these characters. Thus the phrase is less likely to be a meaningful phrase, but a part of a longer phrase.

With the phrase X on an essential node, we assume ***L*** is a set of phrases in the document collection such that each phrase in ***L*** must be in the form bX where b is a single character. For each phrase in ***L***, l_i, the frequency of that phrase is f_{l_i}. The definition of Left Context Dependency (LCD) is

$$|L| < t_1 \text{ or } \max(\frac{f_{l_i}}{f_x}) > t_2$$

where t_1 and t_2 are threshold values.

Similarly, we assume ***R*** is a set of phrases in the document collection such that each phrase in ***R*** must be in the form Xa where a is a single character. For each phrase in ***R***, r_i, the frequency of that phrase is f_{r_i}. The definition of Right Context Dependency (RCD) is

$$|R| < t_1 \text{ or } \max(\frac{f_{r_i}}{f_x}) > t_2$$

where t_1 and t_2 are threshold values.

When the base clusters filtering process is based on the LCD/RCD analysis, we filter out those tentative base clusters that have LCD or RCD. For example, the phrase "董建" is possibly not a good phrase as it is a part of the name "董建華", we expect the phrase "董建" has Right Context Dependency. Another example is the phrase "徒華", the phrase itself does not have meaning but should be part of the name "司徒華". The phrase "徒華" should have Left Context Dependency.

The word analysis using LCD/RCD is an efficient way to filter out the partial phrases that usually contain high dependency on their left or right side. However, a self-contained phrase may not be a meaningful phrase. For example, the phrase "的成" may be self-contained, such that it does not have LCD or RCD, as seen on the phrase "牙痛的成因", "止痛藥的成效", "診所的成本", "電腦化的成果", etc.

Although the phrase “的成” is self-contained, it is not a meaningful base cluster. As we can see, LCD/RCD analysis is good, efficient, and more understandable than SLP analysis, but it is also limited. Therefore, we use another method, CLP analysis that is based on the LCD/RCD analysis, and consider the phrase as in SLP to eliminate phrases with incomplete meaning.

5.1.5.3 CLP Analysis

CLP analysis [16] is a method to determine Complete Lexicon Patterns. The word “complete” means that we are looking for complete phrases in the result. In other words, the CLP analysis aims to remove the phrases with incomplete lexicon meaning. CLP analysis uses the LCD/RCD information together with the mutual information to determine the completeness of a phrase. It assesses whether a phrase is independent of the context and highly associated within the contained text in a certain level.

When we apply the CLP analysis for the base clustering filtering, we apply several criteria in deciding whether a phrase node should be put into result or filter out from the set. We measure its left and right external dependency, as well as its internal association.

With the phrase X on an essential node, we assume $\boldsymbol{R}$ to be a set of phrase in the document collection such that each phrase in $\boldsymbol{R}$ must be in the form Xa where a is a single character. For each phrase in $\boldsymbol{R}$, r_i, the frequency of that phrase is f_{r_i}.

We define X to be Right External Independent (REI) if

$$\begin{cases} |R| \ge t_1 \\ \max(\frac{f_{r_i}}{f_x}) \le t_2 \end{cases}$$

where t_1 and t_2 are threshold values.

Similarly, we assume $\boldsymbol{L}$ to be a set of phrase in the document collection such that each phrase in ***L*** must be in the form bX where b is a single character. For each phrase in ***L***, l_i, the frequency of that phrase is f_{l_i}.

Then, X is Left External Independent (LEI) if

$$\begin{cases} |L| \geq t_1 \\ \max(\frac{f_{l_i}}{f_x}) \leq t_2 \end{cases}$$

where t_1 and t_2 are threshold values.

For the association, we assume Y to be a phrase with length one less than X, such that X is in form of Yc. We also assume Z to be a phrase with length one less than X, such that X is in form of dZ. We define X to be Internal Associated (IA) if

$$\begin{cases} \frac{f_x}{f_y + f_z - f_x} \geq t_3 \\ f_x > t_4 \end{cases}$$

where t_3 and t_4 are threshold values.

With the above definitions, a phrase X is CLP if X is REI, LEI, as well as IA. The definition of LEI and REI is derived from the LCD/RCD analysis. The definition of IA is derived from the SLP analysis. When X is REI, X should be independent from the character to the right of X. So the right boundary of X is a good boundary in certain level. Similarly, LEI is for the left boundary of X. When X is IA, X should be highly associated with its inner contents Y and Z.

5.1.6 Base Clusters Combining

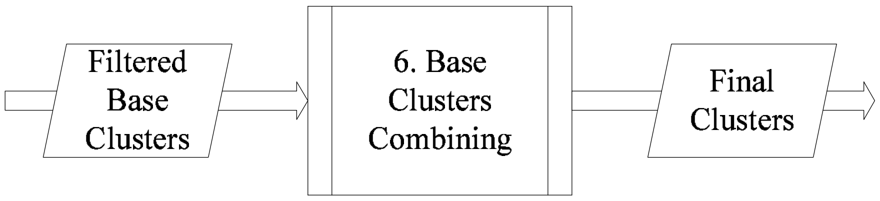


Figure 5.10 Base clusters combining process

The base clusters are the reference for the clustering. It is the center of a cluster. However, some of the base clusters may be highly similar to each other, it is undesirable to treat them as two different clusters with similar contents. Therefore, the grouping process tries to combine the base clusters with high similarity into a single cluster. This process is shown in Figure 5.10.

We use the single-link clustering algorithm to combine the base clusters. This method is similar to the one used in Suffix Tree Clustering (STC) [56]. We denote the set of documents belonging to the base cluster m as B_m and the size of that cluster m as $|B_m|$. When there are two clusters B_m and B_n, with size $|B_m|$ and $|B_n|$ respectively, we merge the clusters B_m and B_n into new cluster if and only if

$$\begin{cases} \dfrac{|B_m \cap B_n|}{|B_m|} > 0.5 \\ \dfrac{|B_m \cap B_n|}{|B_n|} > 0.5 \end{cases}$$

After the above procedure, similar base clusters are joined together and the result forms the final clusters for the set of documents.

5.1.7 Documents Assigning

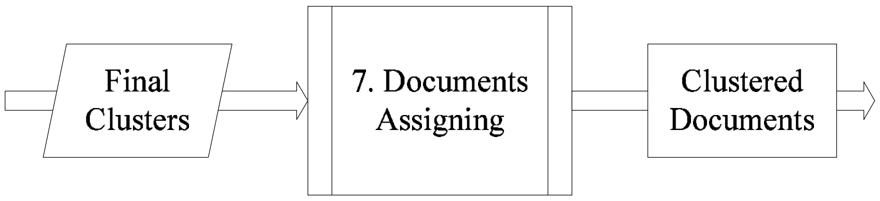


Figure 5.11 Documents assignment process

Up to this moment, the clusters of the document collections are highlighted. We can now assign the documents into the suitable clusters, as shown in Figure 5.11. A cluster should contain as least one phrase for reference. The clusters combined in the grouping process should include more than one reference phrases. We assign a document to the cluster if the document contains all the reference phrases in that cluster. It is possible that a document belongs to more than one cluster.

To make this clusters assigning job efficient, we apply the generic Chinese PAT tree again. We construct a generic Chinese PAT tree for a document. The reference phrases in the clusters can check against the tree efficiently. The process is quite straightforward. Since we can apply the generic Chinese PAT tree for searching of phrases, this cluster assigning process can be done in linear time, provided that the number of discovered clusters is always a finite number.

5.1.8 Result Presentation

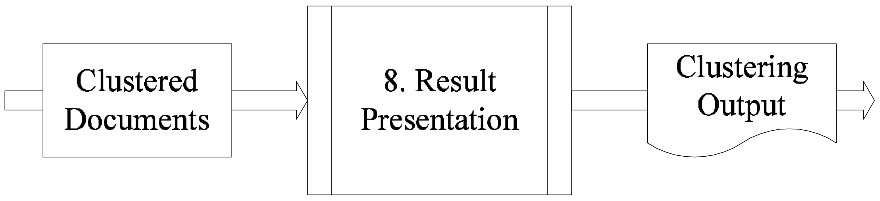


Figure 5.12 Result presentation process

When the documents are assigned to clusters, the final step is the presentation of the clustering result, as shown in Figure 5.12. The result formalizes into expected presentation format. However, the presentation format is application-dependent. It could be the further analysis of the clustering output, the distribution statistics of documents in the clusters, the detailed list of clusters and its documents, etc.

If the clustering of documents is purely for archiving purpose, the clustering result may be used directly for archiving the collection into clusters. In this case, presentation step may be ignored and will be replaced with the proper follow-up action instead.

Since the presentation of result is application-dependent, we leave this part open. In particular, our clustering result can organize documents for browsing on the Web page by making an HTML layout of result displaying the clusters with links to the documents. This kind of presentation is especially good for improving the Web browsing experience on a large collection of non-indexed text.

5.2 Discussion

5.2.1 Flexibility of Our Framework

Our Chinese document clustering framework is a general overview for the clustering of Chinese document collection. Each of the involved steps is well defined with a short-term goal, and the output of each step prepares for the next step. Therefore, it is easy to improve over a single step as long as the target result and requirement does not change.

Our clustering process is adaptive; we can fine-tune each for the processing part so that it is more suitable for a particular application need. For example, the document cleaning process governs the format and style of the raw data. With a suitable documents cleaning engine, the clustering process can accept any kind of documents from the file or from the Web, in pure text format or embedded format. We can select a suitable PAT tree construction process to build our PAT tree. Although we should usually use the generic Chinese PAT tree in the clustering because of its enhancement over the previous design, we still have options on implementations. As we mentioned, dynamic link approach is typical but poor on large document size. Node production factory approach, on the other hand, is very capable for large document with a minor initialization overhead. If the document size is very small in particular, we can select the dynamic link approach or any other approach to build up the generic Chinese PAT tree. In addition, we maintain the option in choosing different ranking functions as the clustering model reference, and we can customize our base clusters filtering method as well as the combining method. All the components are flexible for improvement.

5.2.2 Our Clustering Model

Throughout the research, we try to replace the engine model in different steps. We focus on the clustering of Web-available documents and we target for presentation of results on the Web. In particular, the following engine models are selected in our particular clustering system: (1) we apply the customized cleaning engine for the desired web documents; (2) we use the node production factory approach to handle the construction of the generic Chinese PAT tree; (3) we traverse the tree once to perform the Essential Node detection; (4) we use the weighted frequency as reference to extract the base clusters from the tree; (5) CLP analysis is adopted to filter out the unwanted base clusters from the initial based clusters; and (6) we apply single-link algorithm to combine the base clusters into final clusters.

5.2.3 More About Clusters Detection

It is important to detect the desired clusters for our documents clustering framework. Although our ranking functions consider the phrase ranking based of the frequency, which is a reasonably measurement reference [13], our weight function still has some weakness in certain rare situation. We discuss these and introduce the adjusted weight. We also introduce the artificial stop-word list method that helps improving the clustering result.

5.2.3.1 Ranking by Adjusted Weight

In the base cluster detection, we measure the weight of phrases to determine the set of base clusters. From the general knowledge, a phrase appearing only in one document cannot form a cluster, so it is undesirable for this phrase to get into the base clusters set. However, the ranking function based on the frequency cannot guarantee to avoid this. A phrase with very high frequency in a single document, based on the ranking functions, may still be considered as base cluster. Although there is a rare possibility for this to happen, it is still a possible noise due to the unbalanced high frequency.

To avoid this, we suggest a modifier on the ranking function. We define an adjusted weight ($weight'_p$) for a phrase p, such that,

$$weight'_p = weight_p \times s_p$$

$$s_p = \begin{cases} 1 & \text{if } n_p > 1 \\ 0 & \text{otherwise} \end{cases}$$

where n_p is the number of documents in the document collection containing the phrase p.

When we rank according to $weight'_p$, we can avoid the noise resulted from the phrase that appears only in a single document. Since the value of n_p can be captured form the generic Chinese PAT tree during the PAT tree construction process, this adjustment does not consume much computation time. When the document collection is large, the weight of the real base cluster should be much larger than the noise of single document phrase and the noise will not affect the ranking result. This means the adjusted weight has no effect and it is not necessary to apply the adjusted weight. However, a small set of documents can be very sensitive to these noises. In this case, the adjusted weight becomes an important factor for improving the ranking result.

5.2.3.2 Artificial Stop-word List

In addition to the adjusted weight, artificial stop-word list can improve the base cluster result by filtering out the unwanted phrases in advance. We prepare a list of stop-words that excludes the undesirable phrase to be the base cluster. We take one preprocessing step to mask out the phrases from the stop-word list in the tree. As these phrases are masked out, they are prevented from ranking. Therefore, they can never become the result in the base clusters set.

Common words and domain-specific words are example phrases in the stop-word list. These are not the interesting result for clustering. For example, the phrase “表示” is usually not a significant phrase but it may frequently appear in some type of documents. It is a possible base cluster but not for our interest. Moreover, the phrase “新聞” is a meaningful phrase as base cluster in general, but it would not be interesting when the set of documents is actually a set of newspaper articles. The former example is a common word, and the latter example is a domain-specific word. Artificial stop-word list can prevent them from appearing in the base cluster result.

5.2.4 Analysis and Complexity

Our algorithm is a linear algorithm. We assume that, with a set of *N* documents, the number of words per document is bounded by a constant. Since the construction and searching of a PAT tree is logarithmic to the number of words in the document, the algorithm can perform in linear time. The extraction and grouping processes are similar to those of STC, which are also linear algorithms.

Chapter 6

Evaluations on the Chinese Documents Clustering

We conduct experiments to illustrate that our proposed Chinese document clustering framework is feasible to organize a set of documents into groups. We further investigate the clustering result using the document abstract as well as the clustering result using the document title. The experiment shows that we can cluster the documents with document abstract. Since the abstract is shorter than the original document, this can improve the computation time.

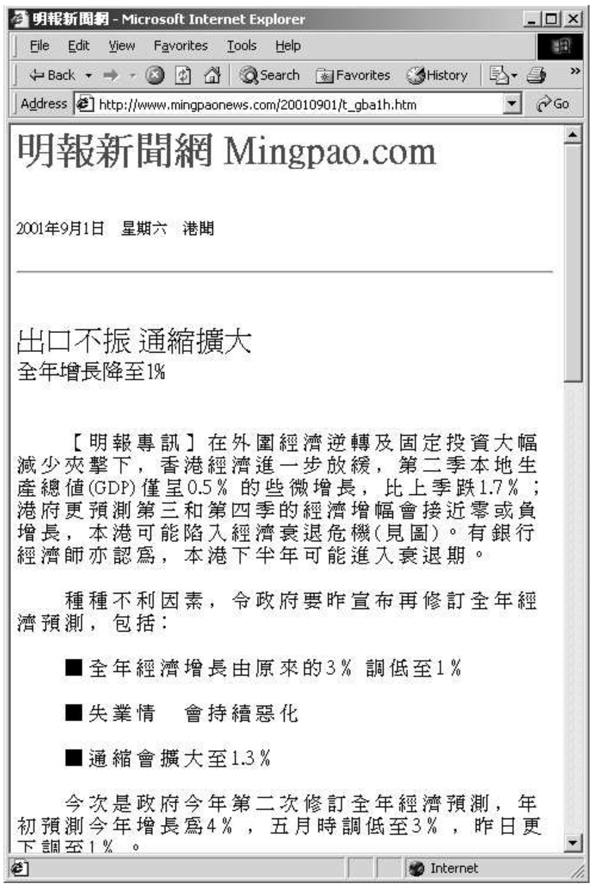
6.1 Details of Experiment

We select the first hundred articles from the local news sections in MingPao News [35] in September 2000 as our documents collection. This collection includes the news from 1st September 2001 to 2nd September 2001. The original articles are HTML documents available on the MingPao News Web site. Figure 6.1a is a sample of the articles, which was published in 1st September 2001.

The documents collection went through the eight steps of our Chinese document clustering process in the experiment. They are 1) Document cleaning; 2)

PAT tree construction; 3) Essential Node extraction; 4) Base clusters detection; 5) Base clusters filtering; 6) Base clustering combining; 7) Document assigning; and 8) Result presentation.

Document cleaning applies to the articles to extract the Chinese content inside. We apply three filters to the original HTML document and obtain the desired cleaned documents. The three filters are: 1) Content filter: It detects the decoration text such as page title, date, time, and copyright footnotes and remove them from the document; 2) HTML filter: It effectively removes any HTML tags or elements from the documents and keeps the plain text information; and 3) Punctuation filter: It strips out the punctuation symbols and any non-Chinese characters and replace them with line breaks. It also handles and suppresses multiple line breaks accordingly. These three filters are incorporated in the Web documents extraction engine. After we collect the documents collection from the Web, those articles are processed and stored in the cleaned form. Figure 6.1b is the sample of the first article in the cleaned form.


明報新聞網 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://www.mingpaonews.com/20010901/t_gba1h.htm
Go
Internet

明報新聞網 Mingpao.com

2001年9月1日　星期六　港聞

出口不振 通縮擴大
全年增長降至1%

【明報專訊】在外圍經濟逆轉及固定投資大幅減少夾擊下，香港經濟進一步放緩，第二季本地生產總值(GDP)僅呈0.5％ 的些微增長，比上季跌1.7％；港府更預測第三和第四季的經濟增幅會接近零或負增長，本港可能陷入經濟衰退危機(見圖)。有銀行經濟師亦認爲，本港下半年可能進入衰退期。

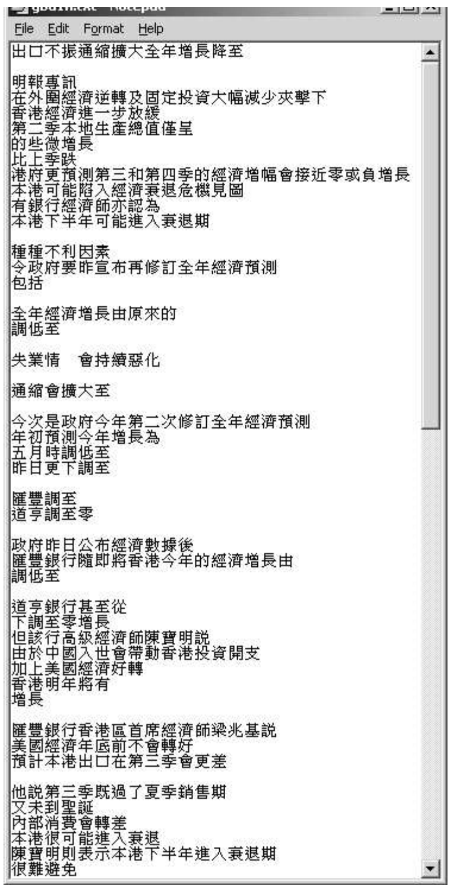
種種不利因素，令政府要昨宣布再修訂全年經濟預測，包括：

■全年經濟增長由原來的3％ 調低至1％

■失業情　會持續惡化

■通縮會擴大至1.3％

今次是政府今年第二次修訂全年經濟預測，年初預測今年增長爲4％，五月時調低至3％，昨日更下調至1％。



```
出口不振通縮擴大全年增長降至

明報專訊
在外圍經濟逆轉及固定投資大幅減少夾擊下
香港經濟進一步放緩
第二季本地生產總值僅呈
的些微增長
比上季跌
港府更預測第三和第四季的經濟增幅會接近零或負增長
本港可能陷入經濟衰退危機見圖
有銀行經濟師亦認為
本港下半年可能進入衰退期

種種不利因素
令政府要昨宣布再修訂全年經濟預測
包括

全年經濟增長由原來的
調低至

失業情　會持續惡化

通縮會擴大至

今次是政府今年第二次修訂全年經濟預測
年初預測今年增長為
五月時調低至
昨日更下調至

滙豐調至
道亨調至零

政府昨日公布經濟數據後
滙豐銀行隨即將香港今年的經濟增長由
調低至

道亨銀行甚至從
下調至零增長
但該行高級經濟師陳寶明說
由於中國入世會帶動香港投資開支
加上美國經濟好轉
香港明年將有
增長

滙豐銀行香港區首席經濟師梁兆基說
美國經濟年底前不會轉好
預計本港出口在第三季會更差

他說第三季既過了夏季銷售期
又未到聖誕
內部消費會轉差
本港很可能進入衰退
陳寶明則表示本港下半年進入衰退期
很難避免
```

Figure 6.1　　a) Original HTML article　　　　b) Cleaned article

With the cleaned documents, the second step of the process constructs the generic Chinese PAT tree of the documents collection. The following third step marks the tree with their Essential Node information.

Step four applies the weighted frequency method on the Essential Nodes of the tree. Each of the Essential Nodes has a weighted frequency value. The Essential Node with a higher weighted frequency value contains Chinese characters segment that is more representative, according to the context of the documents collection. Those with high weighted frequency value become potential base clusters. In this experiment, the threshold value for the lower limit of the weighted frequency value is $weight_{threshold} = (weight_{\max} \times p_w)$, where $weight_{\max}$ is the maximum weighted frequency, and p_w is the parameter of relative lower bound, where $0 < p_w \le 1$. Our experiment takes $p_w = 0.2$ so that, according to the document collection context, the most unimportant Chinese characters segments are grossly excluded.

Step five applies the CLP analysis to further filter the potential base clusters. CLP analysis measures the completeness of Chinese characters segment statistically. With documents collection of reasonable size, CLP analysis can distinguish Chinese word phrases from incomplete characters segments. After the base clustering filtering, the base clusters that contain meaningful Chinese phrases or vocabularies should remain. They become the filtered base clusters.

Step six applies single-link method to combine base clusters with high similarity. We measure the number of documents common to two base clusters: The more the common documents, the higher the similarity. The result after the combination produces final clusters.

Although the number of final clusters must be less than or equal to the number of filtered base clusters, the actual number of final clusters varies depending on the documents collection. To restrict the number of clusters, we can use a threshold value v to obtain the top $v\%$ of clusters, or set a fixed value n to obtain maximum of n clusters. In our experiment, we set $n = 15$ to retrieve a fixed number of clusters.

Documents assignment in the step seven assigns documents to each final cluster. Any document with one or more Chinese phrases in a final cluster belongs to that final cluster. It is possible for a document to become multiple clusters in this documents clustering framework. A cluster is a set of non-trivial documents in the documents collection with topics in common.

Presentation is the final step in the clustering framework to store or present the result. It is of little concern in our experiment. In the result, we present the cluster key of the clusters, which is the Chinese phrases, as well as the cluster size.

6.1.1 Parameter of Weighted Frequency

In the base clusters detection process, the weighted frequency of phrase p is,

$$weight_p = frequency_p \times (baseScore_p)^2$$

$$\text{where } baseScore_p = \max(ChineseCharacterLength_p, topScore)$$

We define the parameter $topScore = 8$ in our experiments. This parameter is a tunable value that can adapt to different situations. It indicates the cut-off length of Chinese phrase we want to emphasize. Under this parameter setup, the range of each $(baseScore_p)^2$ is 1 to 64 inclusively. It emphasizes the Chinese phrases of eight characters or less in quadratic scale, and any longer phrases would not be over-emphasized.

We observe that Chinese vocabularies are usually up to four characters and short phrases are usually within ten characters long. However, phrases with ten or more characters are rare, and the setup of $topScore = 10$ shows an excessive $weight_p$ value for a very long phrase in our early test, which is $\frac{100}{64} = 1.56$ times larger. By restricting the $topScore$ value to 8, the weighted frequency value can highlight short phrases and vocabularies more effectively, and avoid excessive increase.

6.1.2 Effect of CLP Analysis

Our base clusters filtering use the CLP analysis method described in Section 5.1.5.3. CLP analysis is a statistical measurement method that checks Right External Independent (REI), Left External Independent (LEI), and Internal Associated (IA) of phrases in the base clusters. The measurements of LEI and REI analyze the external association of a cluster key. The measurement of IA detects the internal quality of the cluster key.

The phrases of base clusters that pass these checks are regarded as Complete

Lexicon Pattern (CLP). The remaining base clusters are filtered out. In our experiment with one hundred documents in the documents collection, the process maintains the threshold parameters for the CLP analysis as $t_1 = 4$, $t_2 = 0.7$, $t_3 = 0.5$, and $t_4 = 2$.

During the CLP analysis process, base clusters that do not include enough statistical support to become a complete or meaningful Chinese phrase are filtered out from the base clusters set. Table 6.1 is a short list of unwanted base clusters during CLP analysis. They obtain high enough weighted frequency to pass the last detection process. Their key phrases, however, contain little meaning in Chinese. CLP analysis can effectively discriminates them in the base clusters filtering process.

Weighted Frequency	**Base Cluster Key**	**Filtered Reason**
153	經濟增...	Not Right External Independent
144	立法會議...	Not Right External Independent
144	...務司長曾...	Not Left & Right External Independent
135	...港經濟	Not Left External Independent
125	...方接報到場	Not Left External Independent
108	十三歲 (Thirteen years old)	Not Internal Associated

Table 6.1 Examples of removed clusters in the CLP analysis process

For example, the base cluster key “經濟增” is not CLP because it is not REI, which means that the key should have some missing Chinese characters on its right side to produce complete lexical meaning. The base cluster key “港經濟” indicates similar situation because it is not LEI, which means that the key should contain some missing Chinese characters on its left side to present complete lexical meaning. The example like “務司長曾” is neither LEI nor REI, as there should be Chinese characters appended on both sides in order to produce a complete lexicon meaning. These clusters are bad clusters and the CLP analysis filters them out.

The example of base cluster key “十三歲”, which is LEI and REI, is independent of the document context statistically. That key is a Chinese phrase with the meaning of “thirteen years old”. However, it is not IA because that key is a loosely connected word expending from another Chinese word “十三”, which means “thirteen”. Since the analysis detects the poor internal quality, “十三歲” is not CLP in our documents collection. We also filter out that cluster.

After the CLP analysis process, the clusters set contains base clusters with high weighted frequency. Their base clusters key should also contain reasonable lexical meaning. Table 6.2 is a short list of base clusters after CLP analysis.

	Base Cluster Key	**Number of Articles**	**Weighted Frequency**	**Number of Appearances**
1.	經濟 (Economy)	29	452	113
2.	香港 (Hong Kong)	33	344	86
3.	政府 (Government)	27	340	85
4.	被告 (Defendant)	14	288	72
5.	梁錦松 (Antony Leung)	6	270	30
6.	經濟增長 (Economic Growth)	9	256	16
7.	立法會 (Legislative Council)	13	252	28
8.	昨日 (Yesterday)	35	248	62
9.	學生 (Student)	12	244	61
10.	增長 (Grow)	11	212	53
	…			

Table 6.2 Short list of filtered base clusters from a hundred of documents

6.1.3 Result of Clustering

Final clusters are the result after combining the base clusters. Base clusters with high correlation merge together into single clusters. Table 6.3 presents the detail of the top 15 final clusters.

	Final Clusters	Number of Articles
1.	經濟 政府 (Economy, Government)	40
2.	香港 (Hong Kong)	33
3.	被告 案件編號 (Defendant, Legal Case Number)	17
4.	梁錦松 (Antony Leung)	6
5.	經濟增長 增長 (Economic Growth, Grow)	11
6.	立法會 立法會議員 會議 (Legislative Council, Legislative Council Member, Conference)	19
7.	昨日 (Yesterday)	35
8.	學生 (Student)	12
9.	精神上無行爲能力 (Mentally lost of ability)	2
10.	國際刑警 你今次真是揀錯了襲擊對象 (International Cop, You attack the wrong person)	3
11.	警方接報到場 (Police reports to the scene)	5
12.	警方 案件 (Police, Case)	33
13.	西部通道 (West-side pass way)	2
14.	認爲 (Consider to)	26
15.	今年 (This year)	22
	…	

Table 6.3 Final clusters result from a hundred of documents

According to this result highlight, our clustering framework can locate large clusters such as “經濟 政府” (40 out of 100 articles), “香港” (33 out of 100 articles), “警方 案件” (33 out of 100 articles). These are the most representative topics and documents among the whole set of documents collection. Besides, the result detects medium size clusters with significant clusters keys such as “梁錦松” (6 articles), “經濟增長” (11 articles), “立法會 立法會議員 會議” (19 articles). These are well-defined and well-known topics inside our news document collection.

Since we apply an effective CLP analysis process, all the keys of the final clusters are known Chinese phrases or vocabularies. As our framework is based on the statistical analysis of a particular set of documents collection, our result contains clusters with less semantic value such as “昨日”, “今年”. These temporal markers may be essential for some other documents collection, but they are not that important in our news documents collection. To prevent these phrases with no contextual text, we can provide a stop-word list with these phrases. The clustering detection process ignores any words in the stop-word list for being base clusters.

As our document collection is the news articles collection from 1st September 2001 to 2nd September 2001, the obtained clustering result is able to extract the most interesting sets of articles among our selected input documents collection.

6.2 Clustering on Larger Collection

Using the same setup and parameters, we perform another document clustering experiment with a larger set of documents collections. We collect one thousand news articles from 1st September 2001 to 17th September 2001. In this experiment, we expect to extract the most valuable news topics from this upper-half month of September 2001.

6.2.1 Comparing the Base Clusters

Table 6.4 is the intermediate result with the highlights of base clusters just before the base clusters combining process. We can compare this table with Table 6.2 to see the effects of larger set of documents.

The main difference between the two base clusters sets is the magnitude on the weighted frequency. This is due to the difference between the sizes of two documents collections. The phrase in the larger documents set tends to show higher occurrence frequencies, which directly leads to high weight frequency values. Therefore, the weighted frequencies between the different documents collections cannot compare. They are only the relative measurements within their own collection.

	Base Cluster Key	**Number of Articles**	**Weighted Frequency**	**Number of Appearances**
1.	香港 (Hong Kong)	343	4312	1078
2.	政府 (Government)	314	3528	882
3.	美國 (America)	241	3416	854
4.	世貿中心 (World Trade Center)	71	2368	148
5.	經濟 (Economy)	203	2252	563
6.	學生 (Student)	132	2192	548
7.	昨日 (Yesterday)	334	2184	546
8.	政務司長曾蔭權 (Chief Secretary for Administration Donald Tsang)	40	2058	42
9.	公司 (Company)	194	1940	485
10.	學校 (School)	115	1888	472
	…			

Table 6.4 Short list of filtered base clusters from a thousand of documents

Despite the difference on the scale of the weighted frequency, the two sets of documents are from the same source of news articles. Therefore, they contain similar base clusters such as "香港", "政府", and "經濟". This result is natural and it reveals that they are the most popular news topics among them. Besides, all clusters key contains the well-defined Chinese phrase, which is similar to that in the previous experiment.

6.2.2 Result of Clustering

Details of the final clusters result from the set of one thousand documents are listed in Table 6.5. From the result under these documents, the cluster process still detects the most popular topics in the top clusters, namely, “香港” (343 out of 1000 articles) and “政府” (314 out of 1000 articles). This result naturally reveals the fact that most of our news articles are related to these topics.

	Final Clusters	Number of Articles
1.	香港 (Hong Kong)	343
2.	政府 (Government)	314
3.	美國 (America)	241
4.	世貿中心 紐約 世貿 (WTC, New York, World Trade)	133
5.	經濟 (Economy)	203
6.	學生 學校 (Student, School)	177
7.	昨日 (Yesterday)	334
8.	政務司長曾蔭權 曾蔭權 (Chief Secretary for Administration Donald Tsang)	47
9.	公司 (Company)	194
10.	恐怖襲擊 襲擊 恐怖 (Terrorist Event, Attack, Terror)	156
11.	他們 (Them)	247
12.	工作 (Work)	196
13.	認爲 (Consider to)	254
14.	沒有 (Without)	266
15.	停售居屋 居屋 (Suspension of the ownership housing scheme, ownership housing)	46
	…	

Table 6.5 Final clusters result from a thousand of documents

Besides, this clustering result successfully identifies the most important news during the upper half month in September 2001. They are grouped in clusters “世貿中心 紐約 世貿” (133 articles) and “恐怖襲擊 襲擊 恐怖” (156 articles). Another issue “停售居屋 居屋” is also a local hot news topic during that period of time. Our clustering result reports it with 46 related articles. The cluster “政務司長曾蔭權 曾蔭權” (47 articles) is also a hot topics related to the public affair issues of Hong Kong during that time.

This experiment illustrates how the document clustering extract clusters with interesting topics from a set of documents in an unsupervised manner. Final clusters include valid Chinese topics and the most interesting topics are ranked high according to the statistical measurements. However, some final clusters contain common vocabularies like “昨日” or “他們”, which present limited semantic values by themselves. All these clusters contain two characters words as topic. If we do not want them to become clusters in our result, we can avoid them with additional work discuss in the next section.

6.2.3 Discussion

6.2.3.1 Words with Two Characters

From these two clustering experiments with smaller and larger document sets, their results of simple vocabularies, particularly those with two Chinese characters, are quite consistent. Their high ranking is based on their number of occurrences. Since there is no way to evaluate the semantic value of the simple vocabularies in the CLP analysis, those with low semantic values may still be regarded as noise even they are most likely good clusters.

One of the ways to remove them from the result is to simply ignore all two-character words because most of them cannot form clusters of interest. However, some clusters with two-character topic word may still be valuable clusters, like “香港”, “美國”, “經濟”. Since the existence of noise clusters does not affect other good clusters, ignoring all clusters of two-character words becomes undesirable. It harms the clustering result.

On the other hand, we can practically put those unwanted vocabularies into the stop-word list to eliminate them from the final clusters. Stop-word list helps ignore particular words we do not want in the result. The list can be dependent on or independent of a particular type documents collection. We may always provide a list of vocabularies that contain no dedicated semantic values for themselves, such as “沒有”, “但是”, etc. We can also provide on demand a list of undesirable vocabularies for particular type of document collections, such as “今天”, “昨日”, “新聞”, in news collections. Stop-word list is a flexible solution on the documents clustering framework. At the same time, the stop-word list would not affect the results of any other clusters.

6.2.3.2 Clusters Independence

The clusters discovered in our clustering process are independent. Their weighting depends on the statistic value of the cluster key. Therefore, the result of one cluster does not affect the result of another. This property ensures the stop-word list can apply on our clustering framework safely. The words on the stop-word list will be filtered out from the result set, and the result affects no more than these words. The overall result of other clusters is kept as if there is no stop-word list.

6.3 Clustering with Part of Documents

The experiments in this part focus on the clustering result in different variation of documents collection. We use the same set of one thousand documents as our base set, which is identical to the collection set in Section 6.2. We use the clustering result of the base set as the result reference for comparison. In the first variation, we extract only their news headlines to perform the clustering. Since the size of news headlines is much smaller than the size of the whole documents, this experiment perform very fast in processing time. However, the resulting clusters are not outstanding, as much of the valuable result from our base reference does not appear in the top ranks. In the second variation, we extract not only the news headlines, but also the first paragraph of the news articles as news abstract. The clustering on this abstract, although slower than using news headlines alone, still performs much faster than the original experiment. Compared with the result of the base reference, clustering on news abstract can preserve the outstanding clusters. At the same time, clusters with low lexical value tend to rank lower, and clusters with higher lexical value tend to rank higher.

6.3.1 Clustering with News Headlines

In the clustering of news headlines, we use only the headlines of each article in the documents collection. We use the collection of one thousand news articles, which is identical to the collection set in Section 6.2. The clustering process collects text from the headlines of each article to construct the generic Chinese PAT tree.

The headlines of news draw captions and focus of a piece of news articles. We expect that the clustering news result by using news headlines would not be affected much by common vocabularies when compared with the previous result by using full contents of the articles. However, our experiment does not meet this expectation. Table 6.6 is the clustering result by using news headlines of one thousand news articles. These clusters are obviously not as good as those from full document contents shown in Table 6.5.

	Final Clusters	Number of Articles
1.	停售居屋 居屋 (Suspension of the ownership housing scheme, ownership housing)	46
2.	衙門語錄 (Words of the Court)	8
3.	政府徵用線路紐約電話難接通 (Government claims the telephone network. Telephone to the New York was jammed)	2
4.	學生 學校 (Student, School)	177
5.	經濟 (Economics)	203
6.	答市民問 (Reply to Citizens)	5
7.	政府 (Government)	314
8.	回收 (Redeem)	20
9.	內地 (Mainland)	114
10.	港人 (Hong Kong Citizen)	84
11.	世貿 紐約 (World Trade, New York)	133
12.	青年 (Youth)	29
13.	妙問妙答 (Wise question and wise answer)	3
14.	大學 (University)	107
15.	調查 (Investigate)	138
	…	

Table 6.6 Final clusters result from headlines of a thousand of documents

The result of this experiment contains very small size clusters, which is not significant to our documents collection. In the first 15 clusters, four of them are clusters with less than ten articles. They are "衙門語錄" (8 articles), "政府徵用線路…" (2 articles), "答市民問" (5 articles), and "妙問妙答" (3 articles). By inspecting the cluster size and the lexical meaning of the cluster, these clusters are not quite relevant to the documents collection. On the other hand, some representative large clusters, like "香港", "美國", and "恐怖襲擊", in the previous experiment result do not show high score to in ranking at this time. Although clustering with headlines indicates the benefit in reducing clustering time in this experiment, the resulting clusters are far less representative.

From this unsatisfactory result, we further investigate the effect of using news headlines. The reported frequencies of Chinese phrases, which affect ranking of clusters as shown in Table 6.7, provide us some clues. This table illustrates that the number of appearances of Chinese phrase in news headlines is very low. In a thousand of documents, all phrases contain a frequency value less than 30. Consequently, none of the phrases discovered from the news headlines are representative enough for the clustering result. As the range of the number of appearance is only between 1 and 30, the resulting weighted frequency also falls in a limited range. Therefore, the clustering result from the news headlines cannot distinguish the outstanding clusters in its result.

	Base Cluster Key	**Weighted Frequency**	**Number of Appearances in the news headline**
1.	停售居屋 (Suspension of the ownership housing scheme)	176	11
2.	衙門語錄 (Words of the Court)	128	8
3.	政府徵用線路紐約電話難接通 (Government claims the telephone network. Telephone to the New York was jammed)	128	2
4.	學生 (Student)	104	26
5.	經濟 (Economics)	80	20
6.	答市民問 (Reply to Citizens)	80	5
7.	政府 (Government)	76	19
8.	居屋 (Ownership housing)	72	18
9.	回收 (Redeem)	64	16
10.	內地 (Mainland)	56	14
	…		

Table 6.7 Reported frequencies from filtered base clusters of headlines

This experiment demonstrates the usability of our documents clustering framework. It requires large enough document samples to support the statistical measurement in the result. In fact, news headlines are always short sentences with only one or two phrases, and they are not descriptive enough to represent one document. Consequently, the clustering result from using the news headlines alone cannot efficiently categorize the documents collection into outstanding clusters.

6.3.2 Clustering with News Abstract

We define the news abstract as the news headlines together with the first paragraph of the news article. In a piece of news article, the main concept and caption about the news are usually described shortly in its news headlines, and the overview of the whole piece of news is usually presented in its first paragraph. While news headlines are too brief in content for the clustering purpose, we extend our idea to perform the clustering using the first paragraph including the news headlines. The news abstract is the essence of the news. Since the news abstract is usually much shorter than the whole article, the clustering process requires less time to perform on news abstract.

Table 6.8 presents the result of clustering from the news abstract. Comparing with our reference result in Table 6.5, the news abstract provides useful information for the clustering process. In the top 15 results, we observe the large clusters like "香港" (343 articles), "美國" (241 articles), "政府" (314 articles). We also include the outstanding clusters with topics of high lexical values like "政務司長曾蔭權 曾蔭權" (47 articles), "世貿中心 紐約 世貿" (133 articles), "恐怖襲擊 襲擊 恐怖" (156 articles), and "停售居屋 居屋" (46 articles). These are the interesting clusters that we discover in the clustering using the whole documents in the documents collection.

Apart from these interesting topics, results from this experiment show some more clusters with interesting topics that push some 'two-character word' with less lexical meaning into a lower rank. The newly discovered interesting clusters in the top ranks includes "財政司長梁錦松 梁錦松" (37 articles), "行政長官董建華" (21 articles), and "國務院總理朱鎔基" (7 articles). These are famous politicians mentioned in the news on government affairs. The result with these clusters appearing in high rank positively supports the usage of news abstract in our clustering framework. With these three new interesting clusters from this experiment, we further examine the original clustering result from our base reference. The first two clusters appear in the result of the clustering using whole document while the cluster “國務院總理朱鎔基” does not exist there. The two existing clusters were ranked 22 and 43 respectively in the reference clusters result.

	Final Clusters	Number of Articles
1.	香港 (Hong Kong)	343
2.	美國 (America)	241
3.	政務司長曾蔭權 曾蔭權 (Chief Secretary for Administration Donald Tsang, Donald Tsang)	47
4.	政府 (Government)	314
5.	昨日 (Yesterday)	334
6.	世貿中心 紐約 世貿 (WTC, New York, World Trade)	133
7.	恐怖襲擊 襲擊 恐怖 (Terrorist Event, Attack, Terror)	156
8.	財政司長梁錦松 梁錦松 (Financial Secretary Antony Leung, Antony Leung)	37
9.	停售居屋 居屋 (Suspension of the ownership housing scheme, ownership housing)	46
10.	經濟 (Economics)	203
11.	學生 學校 (Student, School)	177
12.	行政長官董建華 (Chief Executive Tung Chee-Hwa)	21
13.	國務院總理朱鎔基 (Chu Yong-Chi)	7
14.	公司 (Company)	194
15.	香港經濟 (Hong Kong Economy)	43
	…	

Table 6.8 Final clusters result from abstract of a thousand of documents

Chapter 7

Conclusion

Chinese information processing technique is an important aspect in the research of Chinese computing. The major challenge of Chinese information processing is due to the nature of ambiguous Chinese word boundary in Chinese documents. Much of the related research efforts are trying to overcome this problem. They mainly focus on topics of keyword extraction, sentence segmentation, and document searching. This thesis is a study of Chinese information processing technique emphasizing the documents clustering problem on Chinese documents collection.

We studied the PAT tree data structure, which is an efficient indexed tree structure in handling sub-strings information of documents, and its variation proposed by Chien for Chinese documents. By identifying the necessary requirements for PAT tree to handle Chinese information and disadvantages of embedded design of Chien's variation, we derived our own variation of Chinese PAT tree.

Based on our variation of PAT tree, we proposed a generic Chinese PAT tree data structure, which benefits from the design of PAT tree and is capable of handling Chinese information. We defined a type of node called Essential Node in the generic Chinese PAT tree. Chinese phrasal information was stored correctly and uniquely inside the Essential Nodes of the generic Chinese PAT tree. Our generic Chinese PAT tree can handle Chinese documents, contains uniform node structure, and is capable of storing correct Chinese phrasal information in the Essential Nodes of the tree.

We identified the large overhead problem for the PAT tree construction. We proposed a node production factory implementation approach to replace the general approach using the dynamic memory management. This solution successfully overcomes the overhead issues. We evaluated the performance of the generic Chinese PAT tree. We showed that construction time the critical process of the PAT tree, and our generic Chinese PAT tree maintained the construction performance in linear time scale, which was comparable to the original PAT tree.

We proposed a system framework for the Chinese documents clustering problem. We applied the generic Chinese PAT tree that we proposed to the clustering framework. The framework is in modular design that can support heterogeneous type of documents and use different Chinese information analysis techniques to perform the detection, filtering, and combining of clusters.

We evaluated the Chinese documents clustering with online news articles from the Web. We applied the weighted frequency ranking criteria and CLP analysis techniques for clusters detection and filtering. The clustering result successfully detect large clusters and clusters with non-trivial topics from the documents collection.

Finally, we evaluated the clustering result with the same collection of news articles using part of the document contents. The result showed that the clustering performs poorly when supplied with only the news headline because news headlines are the over-simplified information for the clustering needs. However, the news headline together with the first paragraph effectively summarizes a piece of news articles. Clustering with this abstract information can perform efficiently and produce result in high quality. Large clusters and clusters with non-trivial topics were contained in the result.

Our Chinese documents clustering framework is an application of our generic Chinese PAT tree. With the satisfactory clustering results and the linear complexity performance, our solution is suitable for the real-time environment such as result clustering systems on Chinese online search engine.

Bibliography

[1] K. Abrahamson. Generalized String Matching, *SIAM Journal of Computing*, volume 16, pages 1039-1051, 1987.

[2] R. B. Allen, P. Obry and M. Littman. An interface for navigating clustered document sets returned by queries. In *Processings of the ACM Conference on Organizational Computing Systems*, pages 166-171, 1993.

[3] Apple Daily Online. *http://appledaily.atnext.com/.*

[4] R. Baeza-Yates and G. H. Gonnet. A new approach to text searching. In *Proceedings of the 12th International ACM SIGIR*, Cambridge, MA, pages 168-175, 1989.

[5] R. Baeza-Yates and G. H. Gonnet. Efficient text searching of regular expressions. In *Proceedings of 16th International Colloquium on Automata, Languages, and Programming*, pages 46-62, 1989.

[6] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*, Addison Wesley, 1999.

[7] L. Baker and A. McCallum. Distributional Clustering of Words for Text Classification. In *Proceedings of the 25th International ACM SIGIR*, Melbourne, Australia, pages 96-103, 1998.

[8] J. S. Chang. Chinese Sentance segmentation Through Constraint Satisfaction and Statistical Optimization. *ROCLING-IV*, Taiwan, pages 147-165, 1991.

[9] A. Chen, J. He and L. Xu. Chinese Text Retrieval Without Using a Dictionary. *SIGIR'97*, pages 42-49, 1997.

[10] K. J. Chen. Word Identification for Mandarin Chinese Sentences. *COLING'92*, 1992.

[11] L. F. Chien. Csmart -- A High Performance Chinese Document Retrieval System. In *Proceedings of the 1995 International Conference on*

Computer Processing of Oriental Languages, ICCPOL'95, pages 176-183, 1995.

[12] L. F. Chien. Fast and Quasi-Natural Language Search for Gigabytes of Chinese Texts. *SIGIR'95*, pages 112-120, 1995.

[13] L. F. Chien and H. T. Pu. Important Issues on Chinese Full-text Information Retrieval. *Computational Linguistics and Chinese Language Processing*, Computational Linguistics Society of Republic of China Press, number 1, volume 1, pages 205-221, 1996.

[14] L. F. Chien. PAT-Tree-Based Keyword Extraction for Chinese Information Retrieval. *SIGIR'97*, pages 50-58, 1997.

[15] L. F. Chien. Exploration of Fundamental Techniques towards Intelligent Chinese Information Retrieval for the Internet. *Institute of Information and Computing Machinery Communication*, number 1, volume 3, 1998.

[16] L. F. Chien. PAT-Tree-Based Adaptive Keyphrase Extraction for Intelligent Chinese Information Retrieval. *Special issue on Information Retrieval with Asian Languages, Information Processing and Management*, Elsevier Press, 1999.

[17] L. F. Chien. Incremental Extraction of Domain-specific Terms from Online Text Resources. *Recent Advances in Computational Terminology*, Ed. By D. Bourigault, C. Jacquemin and M. L'Homme, 2001.

[18] K. Church and P. Hanks. Word Association Norms, Mutual Information, and Lexicography. In *27th Annual Meeting of the Association for Computation Linguistics*, pages 76-83. Association for Computational Linguistics, 1989.

[19] W. S. Cooper, A. Chen and F. C. Gey. Full Text Retrieval based on Probabilistic Equations with Coefficients fitted by Logistic Regression. D. K. Harman, editor, *The Second Text Retrieval Conference (TREC-2)*, pages 57-66, March, 1994.

[20] W. B. Croft. *Organizing and Searching Large Files of Documents*, Ph.D. Thesis. University of Cambridge, October 1978.

[21] D. R. Cutting, D. R. Karger, J. O. Pedersen and J. W. Tukey. Scatter/Gather: a cluster-based approach to browsing large document collections. In *Proceedings of the 15th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 318-329, 1992.

[22] Y. Dai and T. Loh. A New Statistical Formula for Chinese Text Segmentation Incorporating contextual Information. *SIGIR'99*, pages 82-89, 1999.

[23] T. Gaston. New Indices for Text: PAT Trees and PAT Arrays. *Information Retrieval Data Structures & Algorithms*, Frakes and Baeza-Yates (editors), Prentice Hall, pages 66-82, 1992.

[24] G. H. Gonnet, and R. Baeza-Yates. *Handbook of Algorithms and Data Structures in Pascal and C*, 2nd Edition, 1991.

[25] K. K. He, H. Xu and B. Sun. The Design Principal for a Written Chinese Automatic Segmentation Expert System. *Journal of Chinese Information Processing*, volume 5, pages 1-14, 1991.

[26] M. A. Hearst. The use of categories and clustering information access interfaces. In T. Strzalkowski (ed.) *Natural Language Information Retrieval*, Kluwer Academic Publishers, 1999.

[27] D. R. Hill. A vector clustering technique. *Mechanized Information Storage, Retrieval and Dissemination*, North-Holland, Amsterdam, 1968.

[28] H. Kim and S. Lee. A Semi-Supervised Document Clustering Technique for Information Organization. In *Proceedings of CIKM 2000*, pages 30-37, 2000.

[29] K. Kwok, M. Lyu and I. King. A Novel PAT-Tree Approach to Chinese Document Clustering. In *Proceedings of the International Symposium on Information System and Engineering*, pages 85-91, 2001.

[30] A. V. Leouski and W. B. Croft. An evaluation of techniques for clustering search results. *Technical Report IR-76*, Department of Computer Science, University of Massachusetts, Amherst, 1996.

[31] M. Y. Lin, T. H. Chiang, and K. Y. Su. A preliminary study on unknown word problem in Chinese sentance segmentation. *ROCLING*, volume 5, pages 147-176, 1992.

[32] K. T. Lua. From character to word – An application of information theory. *Computer Processing of Chinese and Oriental Languages*, volume 4, pages 304-312, 1990.

[33] K. T. Lua and G. W. Gan. An application of information theory in Chinese sentance segmentation. *Computer Processing of Chinese and Oriental Languages*, volume 1, pages 115-124, 1994.

[34] U. Manber and R. Baeza-Yates. An Algorithm for String Matching with a Sequence of Don't Cares. *Information Processing Letter*, Volume 37, pages 133-136, 1991.

[35] MingPao News. *http://www.mingpaonews.com/.*

[36] D. Morrison. PATRICA-Practical Algorithm to Retrieval Information Coded in Alphanumeric. *Journal of the ACM*, volume 15, pages 514-534, 1968.

[37] J. Nie, M. Briscbois. On Chinese Text Retrieval. *SIGIR'96*, 1996.

[38] T. Ong and H. Chen. Updateable PAT-Tree Approach to Chinese Key Phrase Extraction Using Mutual Information: A Linguistic Foundation for Knowledge Management. In *Proceedings of the Second Asian Digital Library Conference*, Taipei, Taiwan, 1999.

[39] Oriental Daily News. *http://orientaldaily.com.hk/index.html.*

[40] E. Rasmussen. Clustering Algorithms. *Information Retrieval*, pages 419-442, Prentice Hall, Eaglewood Cliffs, N. J., 1992.

[41] C. J. van Rijsbergen. *Information Retrieval*, Butterworths, London, 2nd edition, 1979.

[42] J. J. Rocchio, *Document Retrieval System – Optimization and Evaluation.* Ph.D. Thesis, Harvard University, 1966.

[43] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval,* McGraw-Hill Computer Science Series, McGraw-Hill, New York, 1983.

[44] R. Sproat and C. Shih. A statistical method for finding word boundaries in Chinese text. *Computer Processing of Chinese and Oriental Languages*, volume 4, pages 336-351, 1991.

[45] M. S. Sun and e. al. Some issues on the statistical approach to Chinese Word Identification. *3rd International Conference on Chinese Information Processing*, pages 246-253, 1992.

[46] The Sun Web. *http://the-sun.com.hk/index.html.*

[47] C. H. Tung and H. J. Lee. Identification of unknown words from a corpus. *Computer Processing of Chinese and Oriental Languages*, supplement, pages 131-145, 1994.

[48] E. M. Voorhees. Implementing agglomerative hierarchical clustering algorithms for use in document retrieval. *Information Processing and Management*, Volume 22, pages 465-476, 1986.

[49] P. Weiner. Linear pattern matching algorithms. In *Proceedings of the 14th Annual Symposium on Foundations of Computer Science*, pages 1-11, 1973.

[50] P. Willet. Recent trends in hierarchical document clustering: a critical review. *Information Processing and Management*, Volume 24, pages 577-597, 1988.

[51] Z. Wu and G. Tseng, Chinese Text Segmentation for Text Retrieval: Achievements and Problems. *Journal of the American Society for Information Science*, volume 44, number 9, pages 532-542, 1993.

[52] F. Xu, K. Netter and H. Stenzhorn. MIETTA -- A Framework for Uniform and Multilingual Access to Structured Database and Web Information. In *Proceedings of the 5th International Workshop Information Retrieval with Asia Languages*, pages 41-48, 2000.

[53] Yiming Yang. An Evaluation of Statistical Approaches to Text Categorization. *Kluwer Academic Publishers*, 2000.

[54] C. L. Yeh and H. J. Lee. Rule-based word identification for mandarin Chinese sentences: A unification approach. *Computer Processing of Chinese and Oriental Languages*, volume 2, pages 97-118, 1991.

[55] J. Yen, P. C. Ma, V. Sivakumar and H. Chen. A Software Agent for Analyzing Financial Documents. *Journal of Management Information Systems*, 1999.

[56] O. Zamir and O. Etzioni. Web Document Clustering: A Feasibility Demostration. *SIGIR'98*, 1998.