

PAT Tree and Chinese Keyword Extraction

Chien Chin Chen

Department of Information Management
National Taiwan University

G.H. Gonnet, R.A. Baeza-Yates, and T. Snider, Chapter 5: New Indices for Text: PAT Trees and PAT Arrays, Information Retrieval – Data Structures and Algorithms, Prentice Hall, 1992.

Lee-Feng Chien, "PAT-Tree-Based Keyword Extraction for Chinese Information Retrieval," in Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 50-58, 1997

PAT Tree (1/10)

- The traditional model of information retrieval requires a set of documents.
 - Each document is assigned a list of keywords (terms) with optional relevance weights associated to each keyword.
- This model has some problems, namely:
 - Keywords must be extracted from the text.
 - This task is not trivial and error prone.
 - Queries are restricted to keywords.
- A **PAT tree** is a data structure used to index text collections.
 - Does not use the concept of word.
 - Effective in text processing and search.

PAT Tree (2/10)

- The PAT tree based index method treat the entire text as an array of characters and construct a PAT tree from a set of *sistrings*.
 - A **sistring** (semi-infinite string) is a subsequence of characters from this array, taken from a given starting point but going on as necessary to the right.
- Example of text and sistrings:

Text Once upon a time, in a far away land ...

sistring₁ Once upon a time ...

sistring₂ nce upon a time ...

sistring₃ ce upon a time ...

...

sistring₈ on a time ...

Each sistring is **uniquely** identified by its starting point.

PAT Tree (3/10)

- To make PAT tree easier to understand, we treat text as a sequence of bits.

Text: 0110010001011...

Then

sistring₁: 0110010001011 ...

sistring₂: 110010001011 ...

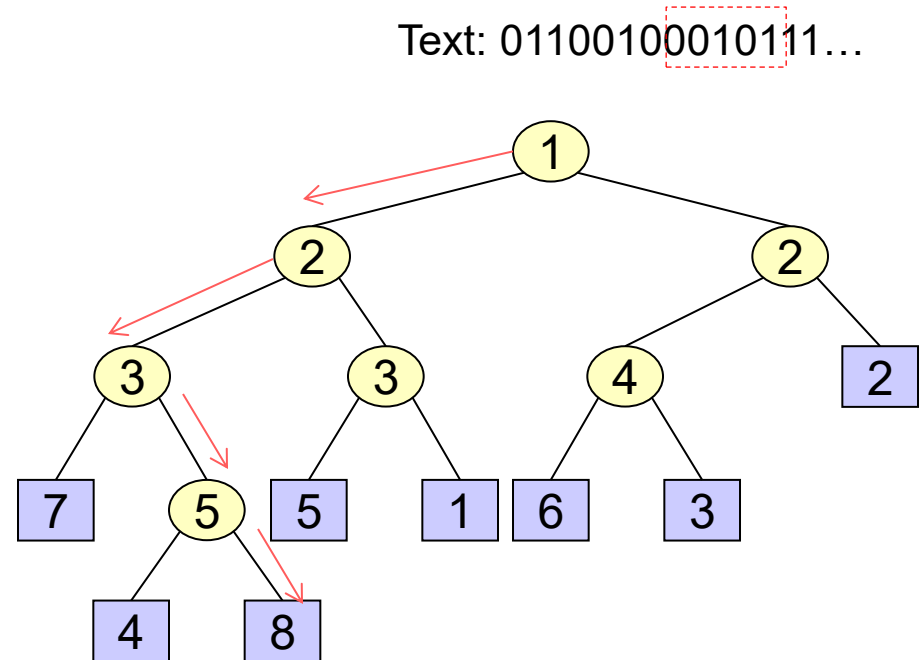
sistring₃: 10010001011 ...

sistring₄: 0010001011 ...

sistring₅: 010001011 ...

PAT Tree (4/10)

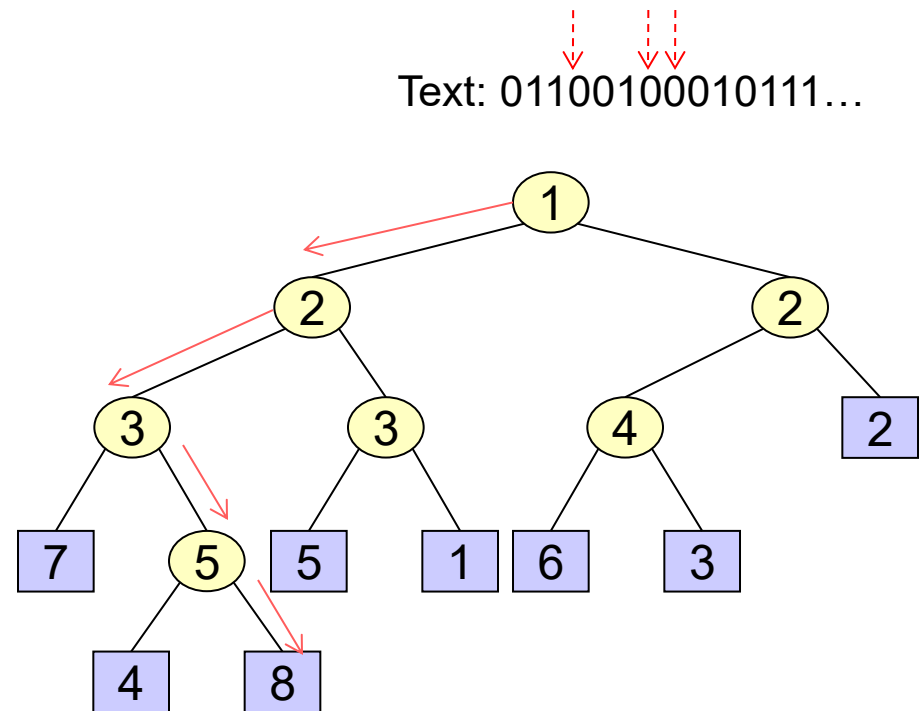
- A PAT tree is a **binary tree**:
 - Constructed over the sistrings of a text.
 - **External nodes** are sistrings.
 - Each **internal node** indicates which bit x of a query is to be used for branching.
 - A zero bit will cause a branch to the left subtree.
 - A one bit will cause a branch to the right subtree.
 - Example: search for 00101.
 - **All the sistrings in a subtree share the same $(x-1)$ prefix.**
 - Some bits (internal nodes) may be skipped.



PAT tree when the sistrings
1 through 8 have been inserted

PAT Tree (5/10)

- Because we may skip the inspection of some bits ...
 - Once we reach our desired node we have to make one final comparison with one of the sistrings stored in the external node of the current subtree.
 - To ensure that all the skipped bits coincide.
 - If not, the needed information is not in the tree.
 - Example: search for 00111.
 - No such information.
 - Example: search for 00
 - Sistrings 4, 7, and 8 are the answers.



PAT tree when the sistrings
1 through 8 have been inserted

PAT Tree (6/10)

- All internal nodes of a PAT tree produce a useful branching.
 - No internal nodes with single descendants.
 - All sistrings in that subtree have the same bit values.
 - Those internal nodes will be eliminated.
- So ... the tree is very compact.
 - For a text of size n ,
 - There must be n external nodes.
 - Correspondingly, there must be $n-1$ internal nodes.
 - The tree (index) size is $O(n)$ – **linear to the size of the text.**

PAT Tree (7/10)

- A **recursive** method for PAT tree construction.

```
Insert(root, sistring)
  if root is NULL
    create an external node representing the sistring.

  else if root is an internal node // root has left and right subtrees
    let x be the bit position of the root
    if the (x-1) prefixes of sistring and a (any) external node
      of root are the same
      if the x bit of sistring is 0
        Insert(root->leftsubtree, sistring)
      else
        Insert(root->rightsubtree, sistring)
    else
      create a parent (internal) node with the branch position and attach
        the sistring and root as its left and right subtrees.
```


PAT Tree (8/10)


```
else // root is an external node
    identify the bit position that branches the two sistrings.
    create a parent (internal) node with the branch position and attach
        the two sistring as its left and right subtrees.
```

PAT Tree (9/10)

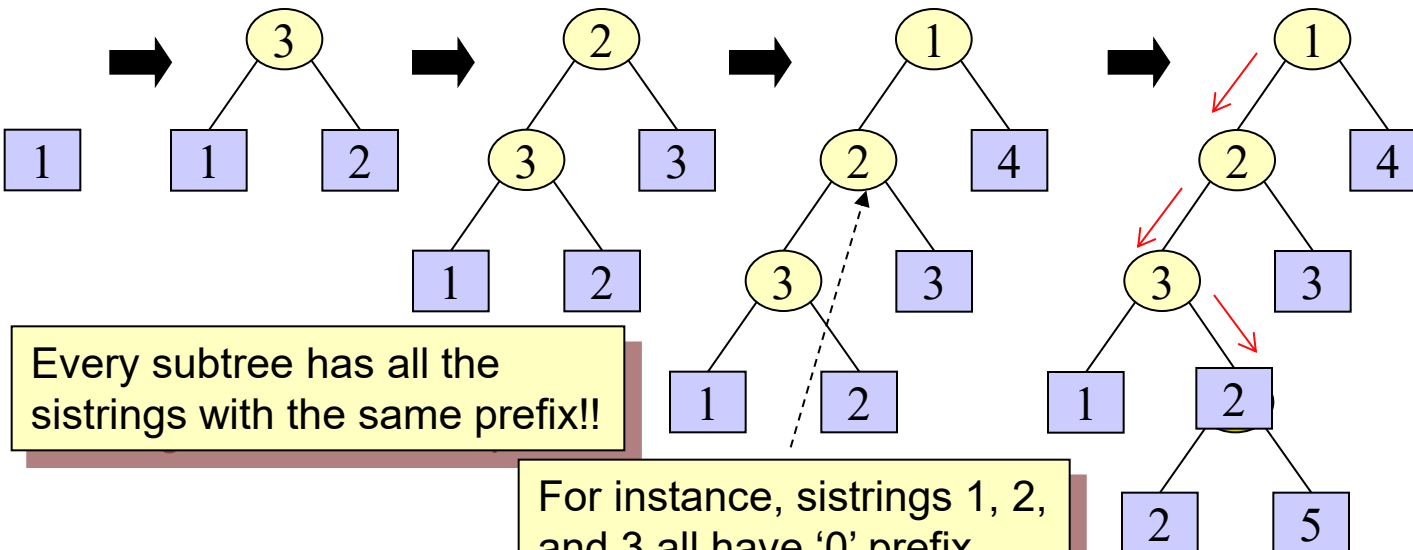
Text: 00010011...

Position: 12345678...

 External node (sistring), labeled by its position

 Internal node, labeled by bit position

root is NULL



An internal node **must** have left and right subtrees.

sistring₁: 00010011...
sistring₂: 0010011...
sistring₃: 010011...
sistring₄: 10011...
sistring₅: 0011...

Every subtree has all the sistrings with the same prefix!!

For instance, sistrings 1, 2, and 3 all have '0' prefix.

PAT Tree (10/10)

□ Implementation details:

- So far we have assumed that every sistring in the text need to be constructed.
- However, **if we are interested in word, then only those sistrings that are at the beginning of words are necessary.**

assuming 8 bits per Chinese character ...

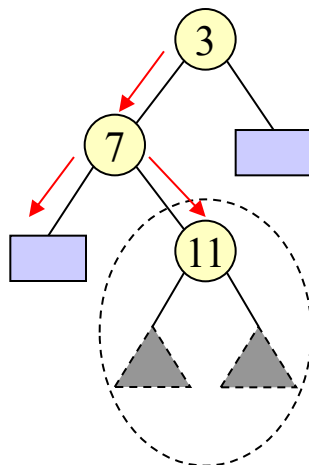
Text:	台	灣	的	...
Bit stream:	00010010011001100110011...			
Position:	1	9	17...	

sistrings needed for indexing

Algorithms on The PAT Tree (1/5)

□ **Prefix searching:**

- Notice that every subtree of the PAT tree has all the sistrings with a given prefix.
- Prefix search in a PAT tree consists of searching the prefix in the tree up to the point where we exhaust the prefix or up to the point where we reach an external node.
 - **At this point we need to verify whether we could have skipped bits.**
 - We compare a (any) sistring in the subtree with the searching prefix.



Prefix search: 00**0**100**1**1 exhaust the prefix
 00**0**100**0**1 reach an external node

The top 10 bits of these sistrings are identical.
But they can be **1**00100**1**.. rather than the searching prefix

If the prefixes match, we can easily identify the locations of the searching information in terms of the sistring positions.

Algorithms on The PAT Tree (2/5)

- For a random PAT tree, the height is $O(\log n)$.
 - **Consequently**, we can do prefix searching in $O(\log n)$ time.
 - **In practice**, the length of the query is less than $O(\log n)$, thus the searching time is proportional to the query length.

- By keeping the size of each subtree in each internal node ...
 - We can trivially find the size of any matched subtree.
 - Knowing the size of the answer is very appealing for some information retrieval purpose.

Algorithms on The PAT Tree (3/5)

□ ***Proximity Searching:***

- Finding all places where a string s_1 is at most a fixed (given by the user) number of characters away from another string s_2 .
- We first search for s_1 and s_2 .
- Then sort by position the smaller of the two answers.
- Traverse the unsorted answer set.
 - Searching every position in the sorted set.
 - Checking if the distance between positions satisfies the proximity condition.

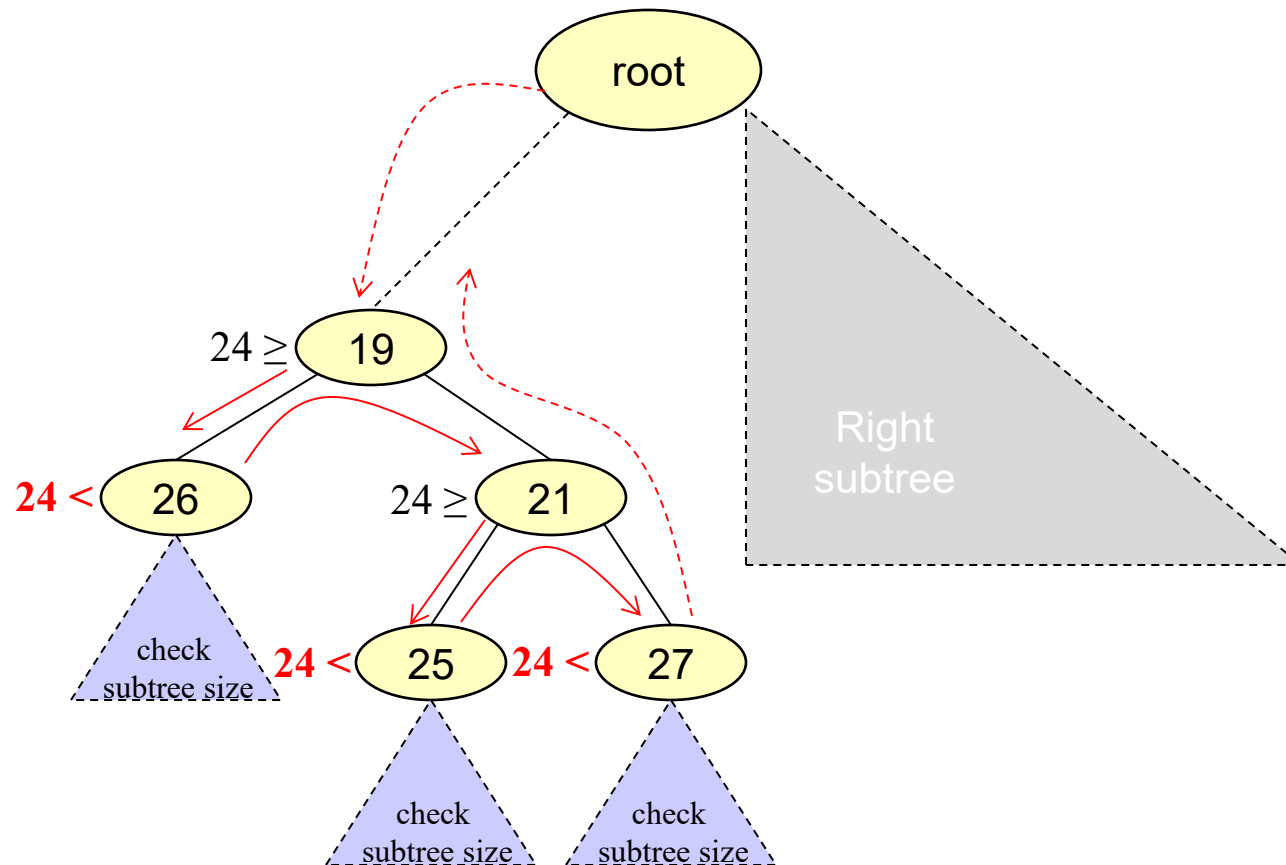
Algorithms on The PAT Tree (4/5)

□ ***Most frequent search:***

- By “most significant” or “most frequent” we mean the most frequently occurring strings within the text.
- Has great practical interest.
- For example, finding the most frequent tri-gram – a sequence of three letters.
 - Is equivalent to finding the largest subtree at a distance 3 characters from the root.
 - Inorder/preorder/postorder traversal.

```
FindMostFreqTrigram(root)
  if root.bit > 24  // assume 8 bit per character
    if # of root's sistring is > max;
      max = # of root's sistring;
      MostFreqTrigram = the first 24 bits of a root's sistring
  else
    FindMostFreqTrigram(root→left);
    FindmostFreqTrigram(root→right);
```

Algorithms on The PAT Tree (5/5)



PAT-Tree-Based Keyword Extraction for Chinese Information Retrieval

Lee-Feng Chien
Institute of Information Science
Academia Sinica
ACM SIGIR 1997

Introduction (1/6)

- The growth in the number of electronic documents in Chinese or other oriental language is enormous.
- These documents are mostly **non-structured** and usually demand efficient IR techniques for retrieval.
- Unfortunately, due to the inherent differences in languages, the techniques developed for retrieving English documents can not be directly applied to these documents.
 - Such as the lack of explicit separators (e.g., blanks) in written oriental sentences to indicate word (term) boundaries.

Introduction (2/6)

- **Automatic keyword extraction** has been a critical problem in Chinese language processing.
- But ... there is not many successful works on Chinese keyword extraction.
- Difficulties of Chinese keyword extraction:
 - **No explicit word boundaries** in written sentences.
 - Too many **unknown words**, such as names, with no effective extraction rules.

Introduction (3/6)

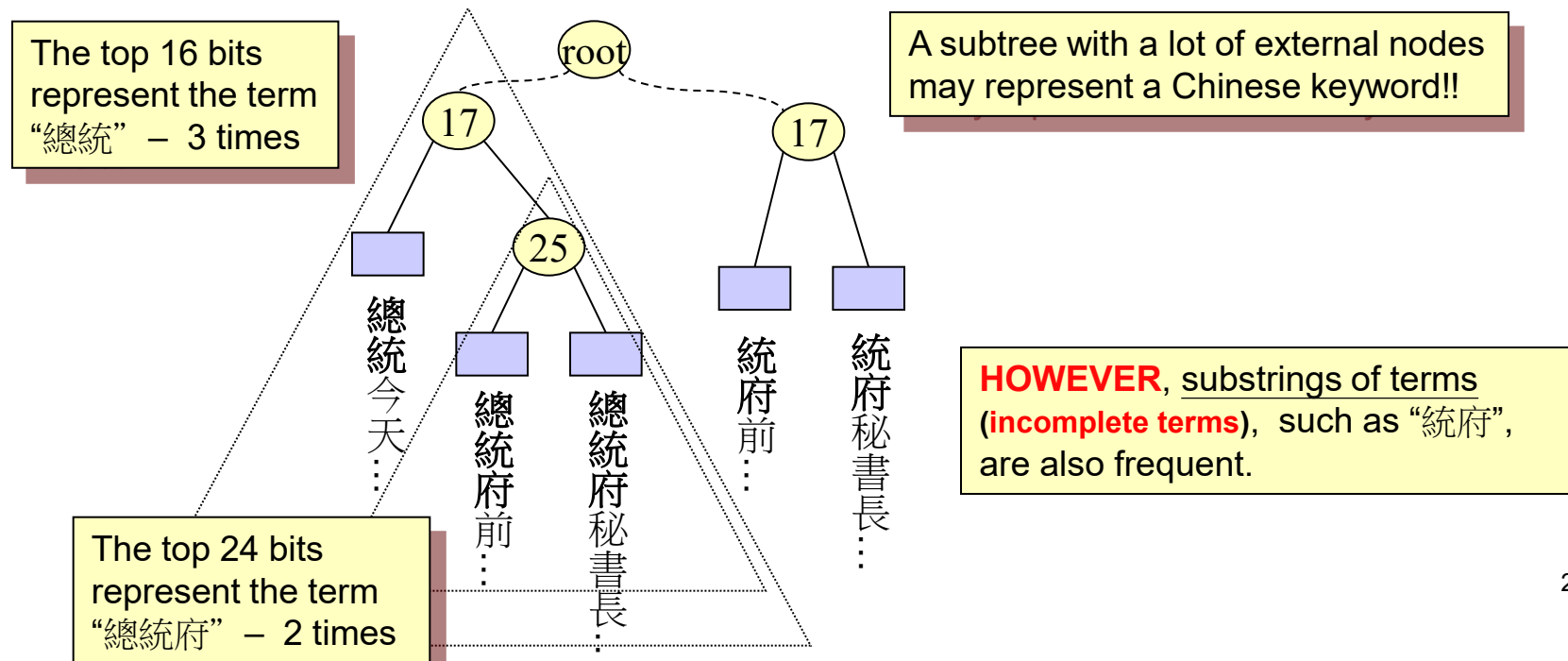
- Traditional ... there are two ways for Chinese keyword extraction:
 1. Ignore the concept of words and use character-level (n-gram) information in the construction of IR systems.
 2. Apply lexicon analysis.
 - Several word-based IR systems rely on a **rigid lexicon** and **sophisticated word segmentation** and **syntactic analysis** in extracting word-level information from documents.
 - Unfortunately ... existing Chinese lexicons are often constructed for general applications.
 - Most proper nouns (such as human names), which are important in describing information need, are often excluded from these lexicon.
 - CKIP (Chinese Knowledge Information Processing) at Academia Sinica, Taiwan, in continuing building a Chinese word lexicon.
 - <http://ckipsvr.iis.sinica.edu.tw/>
 - A milestone construction for Chinese language processing.
 - Contains over 100-thousand word entries (when in 1997).
 - But ... in which only few proper nouns have been included.

Introduction (4/6)

- In this paper, we will not rely on the use of rigid lexicon to determine keywords in the text.
- A keyword ... supposedly ... will occur at least several times in a text collection.
- We try to identify **significant lexical patterns** (SLP) from a set of relevant (Chinese) documents.
 - *Lexical pattern*:
 - Chinese keyword.
 - A string consists of an **arbitrary number** of **successive characters**.
 - For example, “資訊檢索” can be a SLP for a set of IR-related documents.
 - *Significant* → **frequent** and **complete** in semantic.

Introduction (5/6)

- The proposed approach is a **three-step process**.
 1. Using PAT tree data structure to extract all **frequent lexical patterns** from relevant documents.
 - **But ... not every frequent lexical pattern is complete in semantics!!**



Introduction (6/6)

2. **A mutual-information-based filtering algorithm** is then applied to filter out the character strings in the PAT tree which are incomplete in semantics.
3. Finally, a refined method based on a common-word lexicon and a keyword determination strategy are presented.

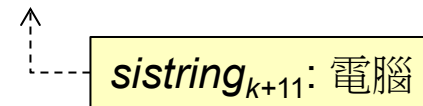
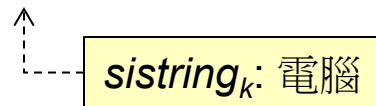
Step 1 (1/3)

- When applying PAT tree to Chinese keyword extraction, instead of recording the sistrings at document level, **we record them at sentence level.**
- We use punctuation marks such as “.” and “,” as delimiters to determine a sentence boundary.
- We logically make sistrings equal length.
 - The length can be the largest size of sentences in the corpus.
 - Short sistrings are inserted “0” bits in the end.
 - So that ... sistrings with different length can be compared.
- Text: 起立，請坐下。
 - sistring*₁: 起立00000000
 - sistring*₂: 立0000000000000000
 - sistring*₃: 請坐下
 - sistring*₄: 坐下00000000
 - sistring*₅: 下0000000000000000

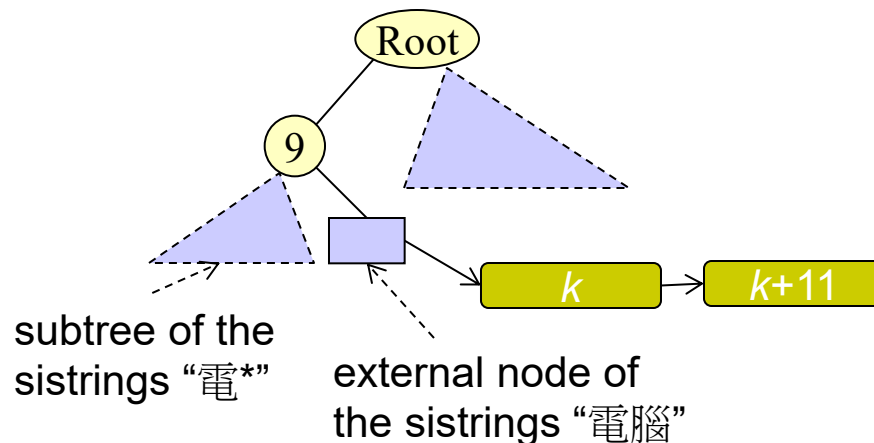
Step 1 (2/3)

- There would be identical sistrings ...

...個人電腦，這是我個人第一台電腦。...



- During implementation, each identical sistring is represented as a node in the PAT tree.
- And has a **linked list** of the sistring positions.



Step 1 (3/3)

- Each internal node consists of three parameters:
 - Comparison bit.
 - Number of external nodes.
 - Frequency of the sistrings.
 - A external node can have more than one sistring.
- By using the frequency information, we are able to know the frequency of any character string in the corpus.
- Then, PAT tree produces a lot of possible lexical patterns.

Step 2 (1/7)

- However, many patterns should be **filtered out**.
 - For instance, if “關鍵字抽取” is **frequent**, then “關鍵字抽” and “鍵字抽取” are **frequent** as well ...
 - However, they are **incomplete in semantics** an lack of representatives.

- According to the findings in our experiments, most of the significant lexical patterns have **strong association between** its composed and **overlapped substrings**.
 - For example – “關鍵字抽” and “鍵字抽取” are highly associated to each other to convey a solid semantics “關鍵字抽取”.

Step 2 (2/7)

- For a frequent lexical pattern $C = c_1c_2c_3...c_n \dots$
 - Its two longest composed substrings, which are $A = c_1c_2...c_{n-1}$ and $B = c_2c_3...c_n$, respectively, must be frequent as well.
 - **But if A and B are highly dependent to C , then A and B should not be complete patterns.**
 - For example, $C = \text{“關鍵字抽取”}$, $A = \text{“關鍵字抽”}$, $B = \text{“鍵字抽取”}$.
- We propose the ***significance estimation function*** (SE) to measure the mutual information of two overlapped substrings.

Step 2 (3/7)

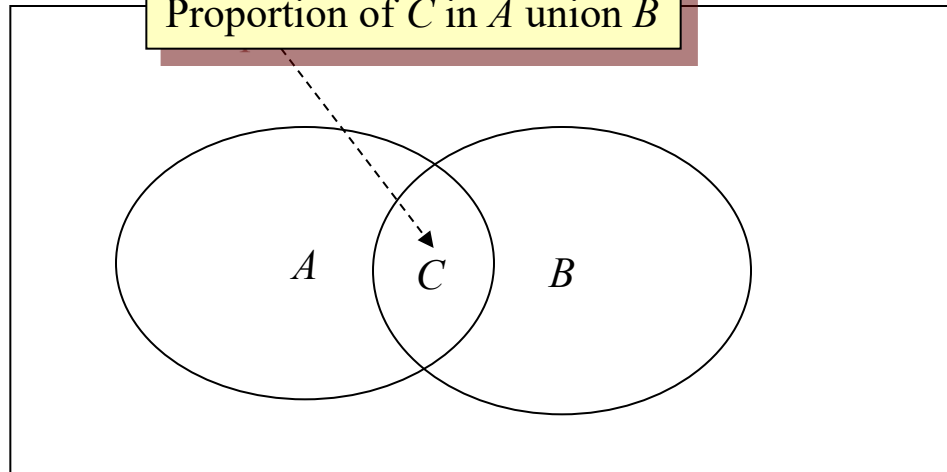
$$SE_C = \frac{\Pr(C)}{\Pr(A) + \Pr(B) - \Pr(C)}$$

frequency of C

$$= \frac{f_C}{f_A + f_B - f_C}$$

High SE_c indicates that A and B are incomplete because they often occur together with C in the text collection.

Proportion of C in A union B



Step 2 (4/7)

- For a **high SE pattern**, we can **filter out** its frequent but incomplete substrings.
- **However, high SE does not indicate that the pattern is complete.**
 - For example, In a text collection, the word “奥林匹克” occurs 6 times, definitely, C = “奥林匹”, A = “奥林”, and B = “林匹”, must occurs no less than 6 times.
 - Suppose that C , A , and B occur exactly 6 times.
 - Then $SE_C = 6 / (6+6-6) = 1$.

Step 2 (5/7)

- On the other hand, **low SE does not** mean an **insignificant** pattern.
 - For example, C = “關鍵字” occur 15 times, A = “關鍵” occur 40 times, and B = “鍵字” occur 15 times.
 - Then $SE_C = 15 / (15+40-15) = 0.375$.
- So ... SE values are only for filtering.
 - To filter substrings of high SE patterns.

Step 2 – Pattern Filtering Algorithm (6/7)

- No offense ... but ... I do not like the algorithm mentioned in the paper.
 - The algorithm is recursive ... and a little bit difficult to understand.
 - And ... may be problematic ... in my opinion ...
- Here is my non-recursive algorithm:
 1. Extract all lexical patterns from PAT tree whose frequency is larger than a pre-defined threshold (TH_f).
 2. Sort the patterns according to their lengths and process them one by one in descending order.
 3. For a pattern C which has not been marked as a *non-SLP*:
 - If $SE_c \geq TH_{se}$ (SE threshold), C 's longest overlapping substring A and B are determined non-candidates of SLP and will have a *non-SLP* mark.
 4. The remaining lexical patterns are **candidate SLPs**.

Step 2 – Pattern Filtering Algorithm (7/7)

$$TH_{SE} = 0.6$$

	關鍵字抽取	10	$SE_{\text{關鍵字抽取}} = 10 / (10 + 10 - 10) = 1$
✖	關鍵字抽	10	
✖	鍵字抽取	10	
	關鍵字	15	$SE_{\text{關鍵字}} = 15 / (15 + 40 - 15) = 0.375$
	關鍵	40	
	鍵字	15	
	...		

- In our findings of the experiments, the above filtering algorithm is especially useful in extracting SLP like names, locations, and technical terms.

Step 3

- There do exist weaknesses with the filtering algorithm.
- A refined procedure, using a lexicon and another set of candidates from a general-domain PAT tree, is employed to remove SLP candidates.
 - If a candidate of SLP appears in the common-word lexicon or in the candidate list of SLP of the general-domain PAT tree ...
 - It will be removed from the list of final SLP.

Experiments – Book Indexing (1/2)

- The experiments were done to extract keywords from a book in the area of IR.
 - Contain 1 MB text and consists of a total about 200,000 Chinese characters.
- The book had been indexed manually first.
 - A total of 190 keywords extracted ... treated as correct answers.
- Since some keywords extracted are similar to manual keywords (with the same meaning but having one or two characters mismatched) ...
 - Both of the **exact** and **near match** method are used in the evaluation.

Experiments – Book Indexing (2/2)

Size of keywords	2	3	4	5	6	>6	Avg.
# of manual keywords	5	10	44	27	46	58	
# of extracted keywords	4	26	106	50	62	28	
# of correct keywords extracted (exact)	2	4	22	14	21	19	
Precision (exact)	0.5	0.15	0.21	0.28	0.33	0.68	0.30
Recall (exact)	0.4	0.4	0.5	0.51	0.45	0.33	0.43
# of correct keywords extracted (near)	2	4	30	19	29	22	
Precision (near)	0.5	0.15	0.28	0.38	0.48	0.79	0.38
Recall (near)	0.4	0.4	0.68	0.70	0.63	0.38	0.56

- Do we have a low precision??
 - 70% of extracted keywords didn't appear in the set of manual keywords.
 - Many of them are actually domain-specific terms and important.
- Do we have a low recall??
 - Most of the keywords missed are due to their low frequency values in this book.

Conclusion

- ❑ Keyword extraction in Chinese is critical and fundamental.
- ❑ The proposed Chinese PAT tree does reduce the difficulty of the extraction task.
- ❑ Keywords, in special proper nouns which were excluded in the general lexicon, are possible to be extracted.
- ❑ Incomplete lexical patterns can be effectively filtered out.
- ❑ Seems great ... but ... in practice, PAT tree really demands large space overhead and takes time to build.
 - Fortunately ... it can be updated (built) incrementally.

Resources & References

- CKIP – a Chinese word lexicon with rigid syntactic information:
 - Chinese Knowledge Information Processing at Academia Sinica, Taiwan.
 - <http://ckipsvr.iis.sinica.edu.tw/>
- Frequent pattern mining – PAT Tree vs association rule.
 - Patterns of PAT tree are consecutive (or sequential).
 - No specific ordering in large itemsets of association rule.
- PAT tree for Information Extraction (IE):
 - Chia-Hui Chang and Shao-Chen Lui, “IEPAD: information extraction based on pattern discovery,” World Wide Web, pp. 681-688, 2001.



You can enhance Chien's work as your term project!!

- Periodically extract Chinese news web pages.
- Using the techniques of the paper to extract Chinese keywords automatically.
- Other crazy ideas ~~~~