(8) 34

FULL-TEXT DATABASE for CHINESE HISTORY DOCUMENT

Ching-chun Hsieh, Zy-Kaan Ding, Yuan-Hua Wang

Shu-Fen Tung, Shi Lin, Chyi-Chou Shu

Computing Center

Academia Sinica

Nankang, Taipei, Taiwan, Republic of China

Abstract

This work is to develop a feasible computer tool or workstation for the humanities studies, especially that there are a lot of full-text data to process. A hybrid access method is applied to meet the goals and a language is designed to describe the full-text data structure and formats.

Up to the present, the database contains almost 5,000,000 Chinese characters(10 Mbytes); and the space overhead is about 20%. Performance shows that structured indexing is suitable in the case of having no specific search subjects.

General Terms: Data Description Language(DDL), Full-text Database, Access Method, Indexing

Additional Key Word and Phrase: Chinese Computer

1. Introduction

Traditional Database Management Systems (DBMS) are designed for formatted records. Some applications, such as Patent Office, Law Studies, Literature Surveys, etc., have large amount of data that are not easy to handle simply by formatting. Full-text processing techniques as mentioned in [FALOUTSOS 1985b][SALTON AND McGILL 1983] should be applied to manage those data.

Chinese documents possess many properties. For example, they appear from top to bottom, right to left. There is no space between characters (words). Therefore commerical products like STAIRS(***), VISTA-FINDER, STATUS, TEXT-TRIEVE are not handled well. The work mentioned here is to develop a software utility (a small subset of database management systems) called "Chinese Text Processor"(abbr. CTP). It manages Chinese full-text data with controlled vocabulary or free-term search and text contents retrieval mechanisms.

This work is to develop a workstation for the humanities studies. It is a part of some long-term projects. The goals involve a full-text database, a set of intelligent tools, such as dictionaries for place, official title, calendar transformation, major events in history, etc.

In the following sections, we will describe our access methods, the data definition language, implementation, performance measurement and future extention as the conclusion.

2. Access method

According to what surveyed in [FALOUTSOS 1985b], there are five different kinds of access methods for text, namely full-text scanning

[AHO AND CORASICK, 1975][BOYER AND MOORE, 1977][HASKIN,1980], inverted file[EMRATH,1983][HOLLAAR, 1978], multi-attribute hashing [AHO AND ULLMAN, 1979][HARRISION,1971], signature file [FALOUTSOS AND CHRISTODOULAKIS, 1984][FALOUTSOS 1985a] and cluster [SALTON AND WONG,1978][SALTON, 1973]. None of them is clearly superior to the others. A very good combination for Patent Office is proposed in [HOLLAR et al,1983]. An excellent system (SMART) is also mentioned in [SALTON, 1971]. In order to choose the best access method, the operational characteristics of the specific environment should be considered, especially for Chinese historical documents.

There are the following characteristics:

(1) Large data bases: excluding tool books, the historical documents of the 24 dynasties alone contain almost 60 million Chinese characters.

(2) Few updates, deletions and insertions.

(3) Access frequency is independent on its ages.

(4) There may be no specific subject to search. Data are just browsed in reviewing a series of books.

(5) The reasonable response time varies with the search space. This will be understood by the users.

Under the above considerations, full inversion of the keywords is not preferred because there are too many overheads. That is why three kinds of access methods are applied, combined with full-text scanning, structured indexing and inverted list techniques.

2.1 Full-text scanning

For most free-term searches, the key-table contains multiple keywords and is sorted by their internal code order [KNUTH,1973]. The search space is based on a 5-tuple structured index element (path, W-JUST, W-LETF,

W-RIGHT, W-NEXT). The complexity is O(mlogk) where m is the search space decided by the structured index elements, and k is the number of keywords in the sorted key-table.

2.2 Document-structure indexing

It is mentioned in [PEELS et all, 1985] that documents have their "natural structures" such as chapters, paragraphs, titles, figures, etc. The above logical meaning is analysed by DDL to form a tree-like hierarchica model. (The DDL will be discussed in the next section and Appendix A) Figure 1 shows the logical meaning of the documents. Figure 2 depicts the physical meaning of the documents. This indexing is very suitable for some retrievals of browsing the data in very natural ways.

The 5-tuple structured index element (path, W-JUST, W-LEFT, W-RIGHT, W-NEXT) is designed as follows:

path: indicates path no. from root.

W-JUST: current node.

W-LEFT: the leftmost succeeding child node.

W-RIGHT: the rightmost succeeding child node.

W-NEXT: next node to be traversed.

This element does not only act as guide to traverse through the structure but also decides the search space(2.1).

2.3 Inverted list

For some specific subjects, for example, ecomonic history, controlled vocabulary keyword search is provided. Keywords in ecomonic history are classified into general categories (e.g. ancient person name, place, dynasty etc.). The terms are just inverted and sorted. A small set of cross-

reference for ancient person names (related-term structure [SALTON, 1986]) can be provided so that there will be a greater recall rate. The cross-reference access is achieved by hashing and conflict resolution by rehashing.

3. DATA-DEFINITION LANGUAGE(DDL)

According to SGML([ISO/DIS 8879,1985] Standard of General Markup Language), a context free grammar is defined. In Appendix A, a BNF representation of the structurization parser is presented. This BNF is modified from [HSIEH et al, 1986]. The mark-up language describes the natural structure of documents in addition to ordered list, see-also reference, keyword, and font size. Attempts are made to develop a generalized markup language for Chinese text. The language is not only suitable for historical documents but also for law, newspaper, government documents, etc.

4. Implementation

This project namely " Automation of Historical Literature " has been launched since July 1st. 1984. Currently, there are three major versions of the CTP's on different systems. Table 1 lists their characteristics and highlights.

4.1 CTP 1.0 (1985)

It is implemented in the IBM PC (model 5550 with a 10 Mbyte hard disk). A specific area of historical documents has been collected (economic behavior in five dynasties). The controlled vocabularies (keywords) are classified into six catgories, namely name, place, time, offical title, reference and major events. Therefore 6 inverted list files are built to include these

keywords and the pointers of the corresponding basic text elements.

The name inverted list file associated with an ancient name cross-reference file will induce a greater recall rate in related-term keyword sea To access the cross-reference file, a hashing function based on division techniques is applied. The collision probability is less than 0.1%. Conflict resolution is archived by another hashing function (rehashing).

The data structures and the program flows will not be discussed in more detail(Reference can be made to [MAO,1985]). There are friendly manual operations provided in the CTP. The full-text data can be displayed in horizontal or vertical format as required. The keywords being searched are highlighted.

4.2 CTP 2.0 (1986)

This is the first multi-user version of the CTP. The whole data structures based on those of the CTP 1.0 . In this version, the full-text scanning access method is applied as an experiment and twice the amount of data are tested in micro-VAX II under the VMS 4.1(*) environment. There are also manual operations as a guide in the CTP 2.0 . But the system prompt messages are in English instead of in Chinese.

There is an installation of this version is in the East-Asia Library, University of Washington, Seattle, Washington, U.S.A.

4.3 CTP 3.0 (1987)

The CTP 3.0 is implemented in AT&T 3B15 under UNIX V(*)/BINIX(**).

The coding language is C. The current amount of data is about 5,000,000 Chinese characters, equivalent to 10 Mbytes. The document-structurd index file size is about 2 Mbytes. The basic text element handled is a paragraph.

There are two major processes in the CTP 3.0: one is the Creation Module(CM) and the other is the Retrieval Module(RM). Although the whole system is integrated as shown in figure 3, both modules can work independently.

4.3.1 Creation Module(CM)

There are two major functions in the CM. The first one is a data-entry functional module. It converts the original text into a machine readable form called source text file. The source text files are processed by a text structurization module (DDL parser). The second function produces the whole database which contains document-structure index files, inverted list files, text string files and format files.

The functional block diagram of the CM is shown in figure 4. The document-structure index file contains structured index elements as mentioned in section 3.2. The inverted list file contains classified subject headings if this is specified in the source text files when the database is created. The text string file keeps the original text string and the format file stores all type-setting information of the original text, such as page number, font size, etc.

To build the structured index tree is the most important function of the structurization module. Our approach is to design a nonrecursive predictive parser[AHO, SETHI and ULLMAN, 1985]. This parser is a special case of the recursive-descent parsing. It creates the tree nodes of the

structured index parse tree in preorder. The average depth of the tree is 6 and the total amount of leaf nodes is 20,000 .

Some text elements could be recognized by default without being marked. For example, a paragraph is identified by two blank characters at its beginning and the ordered list is identified by the special leading characters " [". The mark-up symbols of the level structure for the paragraph or ordered list may be omitted without affecting the function of the parser. This strategy saves a lot of effort in marking.

4.3.2 Retrieval Module(RM)

A functional block diagram of the RM is shown in figure 5. There are two functions in this RM. One is the tree traversal module. The other is the list chain module. The tree traversal module loads tree nodes (the 5-tuple structured index elements) on demand basis. At the beginning, the 5-tuple structured index element of the root node will decide which succeeding nodes are going to be loaded. Then each loaded node will decide its succeeding nodes again until the goal nodes have been visited.

When the available workspace is fully occupied, the second chance replacement algorithm[PETERSON and SILBERSCHATZ, 1982] is used as the load-replacement strategy.

The function of the list chain module is to find proper tree nodes according to the resolution of the query understanding module. The paragraphs and their ordered list (see-also references or foot-notes) pairs could be shown in display alternatively by a single keystroke. The query understanding module processes friendly manual operations.

5. Performance measurement

5.1 CTP 3.0

In the case that there are no specific subjects or keywords to retrieve, the system will prompt with the results immediately after the return key is struck (the default is to show the next paragraph) as the book is being reviewed.

The system turn-around time for retrieval of a series of free term keywords is dependent on the number of keywords and on the search space. The time against keyword terms is plotted in Figure 6.

5.2 CTP 2.0

The average turn-around time for retrieval of a single keyword with a search space of 400 Kbytes is about 40 seconds. The average result for the inverted list access method is about 1 second.

5.3 CTP 1.0

In most typical areas of economic history, data contain almost 200,000 Chinese characters. The inverted list access method is applied at most times (inverted list file size is about 35 Kbytes). The average result for single controlled keyword queries, is about 1 minute.

The major reason for this slow response time is that this version of the CTP is implemented in BASIC interpreter.

6. Conclusion and further applications

(1) The structured index is suitable for historical literature processing for beginners.

(2) For performance improvement, some alternative free text scanning techniques[AHO AND CORASICK, 1975] may be adopted to justify Chinese keyword properties.

(3) The standard mark-up language for text is necessary especially for multi-attribute documents.

(4) Statistical Analysis for Chinese character/word usages is the most proper application of the CTP.

(5) The development of full-text database and full-text processing technology provides with an excellent opportunity to integrate information retrieval systems, text/word processing systems, and computational/analytical systems all together.

7. Acknowledgements

The authors wish to thank P.S. Ting, H.K. Mao, K.J. Chen, S.S. Tseng, K.Y. Tam, P. Haiso, J.Y. Liu, Y.C. Ho and P.Y. Huang for their advice or thoughtful reading or typing of this paper.

This project is financially supported by the Committee of Cultural Creation, Executive Yuan, R.O.C. for three years (1985-1987). The authors also want to thank the Data Entry Group of Computing Center, Academia Sinica, R.O.C. for their editing work on the source text files.

***

STAIRS	is registered by IBM.
VISTA-FINDER	is registered by DATAPOINT.
STATUS	is registered by COMPUTER POWER.
TEXT-TRIEVE	is registered by BURROUGHS.

***

BINIX	is a trademark of SERTEX; stands for BIlingual-uNIX.

*

UNIX	is registered by AT&T.
VMS	is registered by DIGITAL.

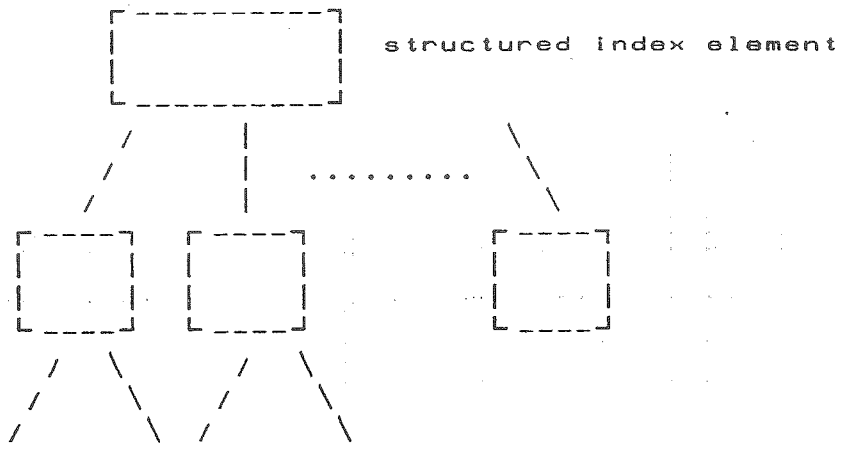

Figure 1. Logical meaning of the documents

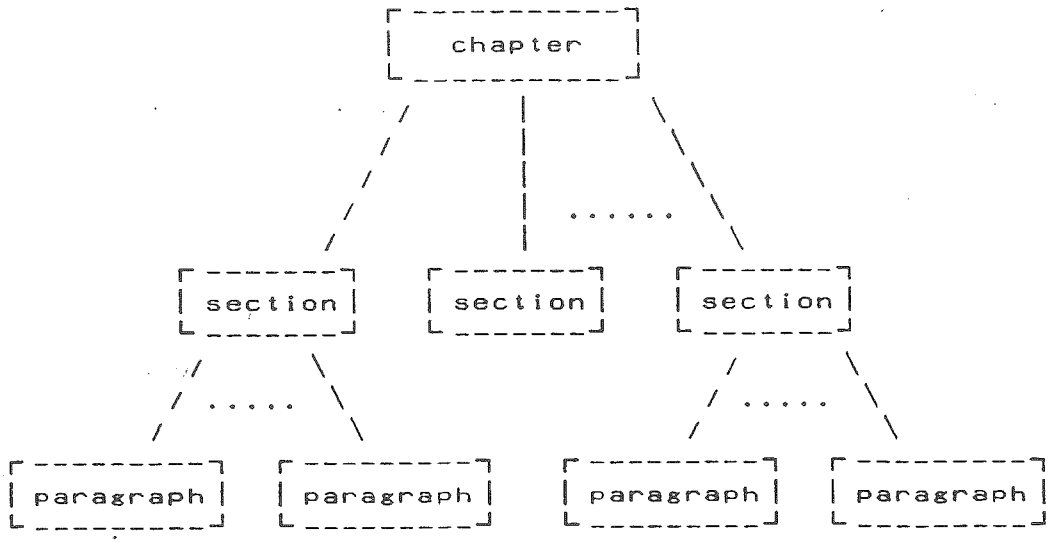

Figure 2. Physical meaning of the documents

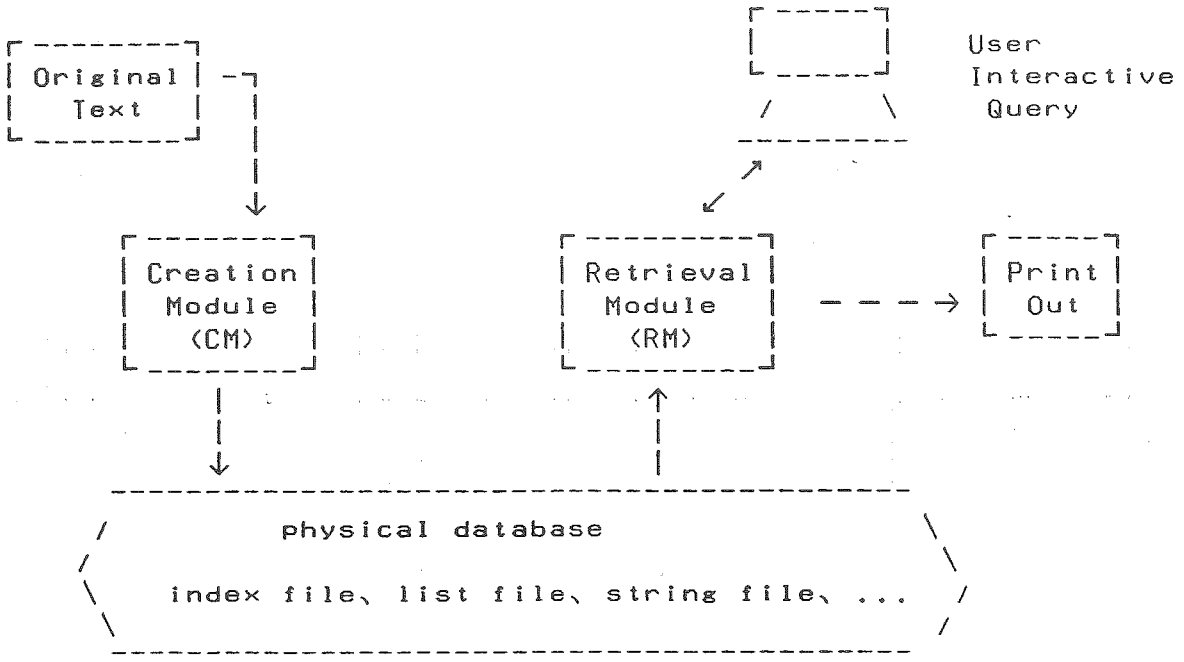

Figure 3. CTP Functional diagram

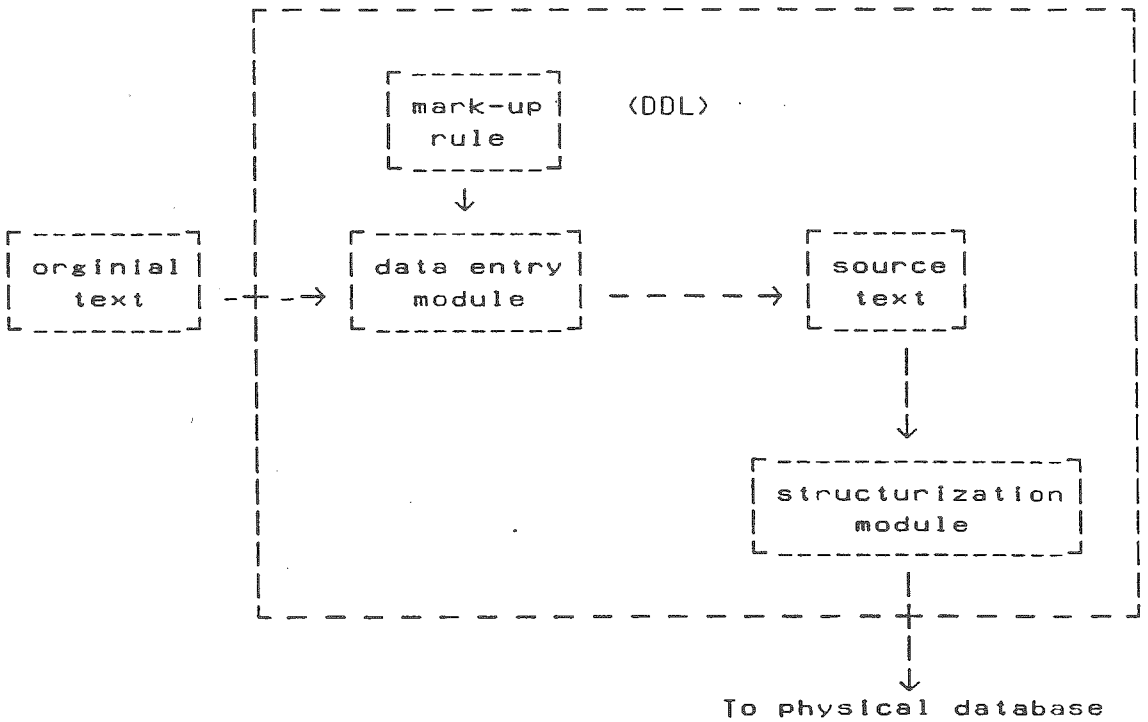

Figure 4. A block diagram of the creation module

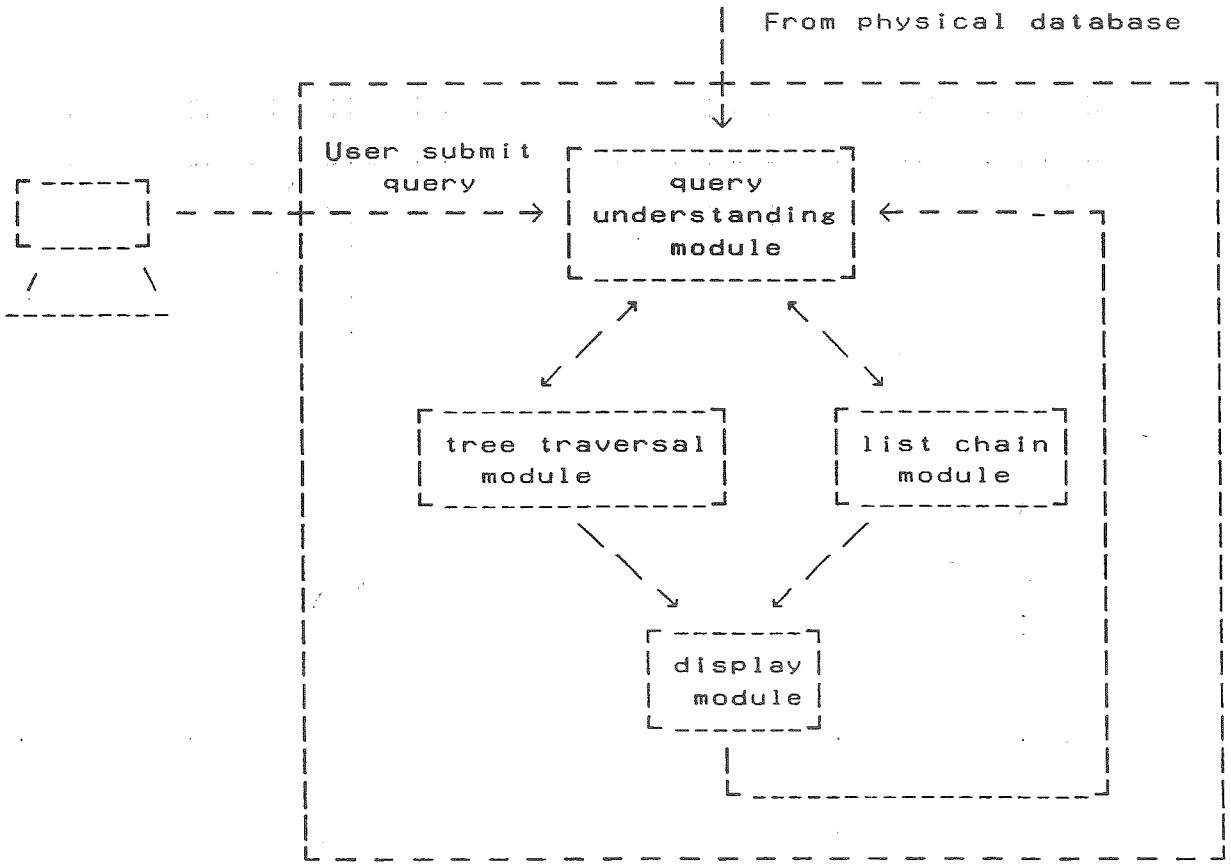

Figure 5. A block diagram of the retrieval module

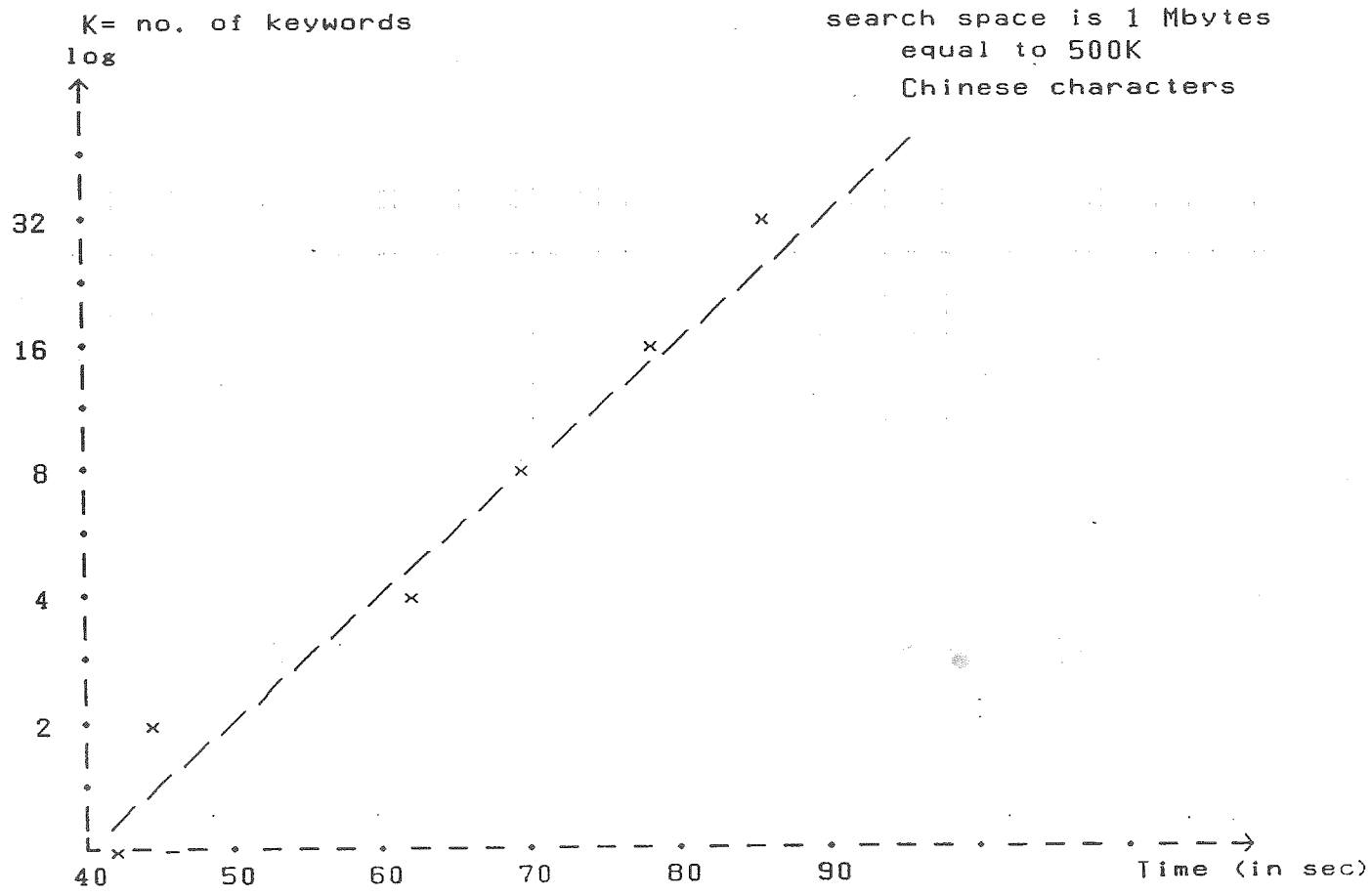

Figure 6. Full-text scanning turn-around time measurement

	CTP 1.0	CTP 2.1	CTP 3.1
1. Hardware	IBM 5550 (16-bit PC)	MICRO-VAX II with Dragon 570 Terminal	AT&T 3B15 with Dragon 570 terminal
2. Operating System	PC-DOS	VMS 4.1	UNIX V BINIX 1.0
3. Language used	BASIC (Intepreter)	C	C
4. Query Form	Interactive	Interactive	Interactive
5. Basic text element	Page-Paragraph	Page-Paragraph	Paragraph
6. Access Method	Inverted list	Inverted list Full-text Scanning	Document-structure Indexing Full-text Scanning
7. DDL	none	none	applied
8. Text size (in bytes)	400K	1200K	10M
9. Overhead size (in bytes)	35K	105K	2M
10. No. of Users	1	2-4	8-48
11. Date of Issue	March,1985	March,1986	Sept.,1987
12. Place of Installed	Computing Center Academia Sinica Taiwan,R.O.C.	East-Asia Library Univ. of Washington Seattle,U.S.A.	Computing Center Academia Sinica Taiwan,R.O.C.

Table 1. The characteristics of different versions of CTP

References

AHO, A. V. AND CORASICK, M. J.
1975 Fast Pattern Matching : An Aid to Bibliographic Search.
CACM 18, 6(June), pp333-340

AHO, A. V., SETHI, R. AND ULLMAN, J. D.
1985 Compilers,Principles,Techniques, and Tools.
Addison-Wesley Publishing Company, Mass., Ch.2-4

AHO, A. V. AND ULLMAN, J. D.
1979 Optimal Partial Match Retrieval When Fields are
Independently Specified.
ACM Tr. On Database System 4,2(June), pp168-179

BOYER, R. S. AND MOORE, J. S.
1977 A Fast String Searching Algorithm.
CACM 20, 10(Oct.), pp762-772

EMRATH, P. A.
1983 Page Indexing for Text Information Retrieval Systems.
Ph.D. Dissertation, UI at UC, May.

FALOUTSOS, C., AND CHRISTODOULAKIS, S.
1984 Signature Files : An Access Method for Documents and Its
Analyitcal Performance Evalution.
ACM Tr. on Office Inf. System 2,4(Oct.), pp267-288

FALOUTSOS, C.
1985a Signature Files : Design and Performance Comparison of Some
Signature Extraction Methods.
In Proceedings of The ACM SIGMOD Conference (Austin, Tex.,
May).
ACM, pp63-81

FALOUTSOS, C.
1985b Access Methods for Text.
ACM Computing Surveys 17,1(March), pp49-74

HARRISON, M. C.
1971 Implementation of the Substring Test by Hashing.
CACM 20, 10(Oct.), pp777-779

HASKIN, R. L.
1980 Hardware for Searching Very Large Text Database.
In Workshop on Computer Arch. for Non-Numeric Processing.
pp49-56

HOLLAAR, L. A.
1978 Specialized Merge Processor Networks for Combining Sorted
Lists.
ACM Tr. on Database System 3,3(Sept.),pp272-284

HOLLAAR, L. A., SMITH, K. F.,CHOW, W. H., EMRATH, P. A.AND HASKIN, R. L.
1983 Architecture and Operation of A Large, Full-Text information
-Retrieval System.
In Advance Database Machine Architecture, D. K. Hsiao, Ed.
Prentice-Hall, Englewood Clitts, N. J. Chap. 9.

HSIEH, C. C. et al.
1986 The Chinese Full Text Processing
(Technical Report in Chinese)
Academia Sinica, R.O.C.

ISO/DIS 8879,
1985 Information Processing-Text and Office System-Standard
Generalized Markup Language(SGML).
Draft Internation Standard. ISO

KNUTH, D. E.
1973 The Art of Programming. vol. 3 : Searching and Sorting.
Addison-Wesley Publishing Company, Mass.

MAO, H. K.
1985 The Report of History Log Automation
(Technical Report in Chinese)
Academia Sinica, R.O.C.

PEELS, A. J. H. M., JANSSEN, N. J. M. AND NAWIJN, W.
1985 Document Architecture and Text Formatting.
ACM Tr. On Office Information System 3, 4(Oct.), pp347-369

PETERSON, J. L. AND SILBERSCHATZ, A.
1982 Operating System Concepts.
Addison-Wesley Publishing Company, Mass.

SALTON, G.
1971 The SMART Retrieval System-Experinments in Automatic
Document Processing.
Prentice-Hall, Englewood Cliffs, N. J.

SALTON, G.
1973 Recent Studies in Automatic Text Analysis and Document
Retrival.
JACM 20, 2(April), pp258-278

SALTON, G. AND WONG, A.
1978 Generation and Search of Clustered Files.
ACM Tr. on Database System. 3, 4(Dec.), pp321-346

SALTON, G. AND McGILL, M. J.
1983 Introduction to Modern Information Retrieval.
McGraw-Hill, New York, N. Y.

SALTON, G.
1986 Another Look at Automatic Text-Retrieval Systems.
CACM 29, 7(July), pp648-656

Appendix A

1. Syntax description

(1) Meta-symbols of the BNF Notation

< > ::= | { } [ ] @

(2) Constructs of the BNF Notation

<..> <..> Angular brackets enclose syntactic constructs denoted in English words and in sequence express concatenation.

.. ::= .. This symbol means "is defined as".

.. | .. A vertical line expresses choice, to be read as "or".

{ .. } Curly brackets express repetition, that is,

{a} = <empty> | a | aa | aaa | ...,

where <empty> denotes the null sequence of symbols, the empty string.

[..] Square brackets surround optional elements, that is,

[a] = <empty> | a.

.. Terminal symbols should not be enclosed between the angular brackets.

@ABC Markup symbol.

2. Markup symbol

(1) Basic element

< alphabet > ::= a | b | c | .. | z | A | B | C | .. | Z

< digit > ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

< ASCII special symbol > ::=

! | " | # | $ | % | & | ' | (|) | * | + | , | - | . | / |

: | ; | < | = | > | ? | @ | [| \ |] | ^ | _ | ` | { | } |

~ | < bar >

< bar > ::= |

< special symbol > ::=

< ASCII special symbol > | < machine dependent special symbol >

< Sino-character > ::=

character set

< letter > ::= < alphabet > | < Sino-character >

< form effector > ::= \ n

< char > ::=

< letter > | < digit > | < special symbol > | < form effector >

< id > ::= < letter > { < letter > | < digit > }

< unsigned integer > ::= < digit > { < digit > }

< string > ::= " < char > { < char > } "

(2) Declaration

<concrete symbol declaration> ::=

#concrete

<concrete statement> { <concrete statement> }

#end

<concrete statement> ::= <system markup symbol> =

<concrete symbol>;

<system markup symbol > ::= <id>

<concrete symbol > ::= <id>

```
<ordercharacter> ::=
    #orderchar : <code char type>
      charset = ( <enum char> { , <enum char> } ) ;
    #end
      <code char type> ::= <unsigned integer>
      <enum char> ::= <alphabet> | <digit> | <Sino-character>

<ordercode> ::=
    #ordercode : <code type>
      <orderpr>
    #end

      <code type> ::= <unsigned integer>
      <orderpr> ::=
        chartype = <code char type> ; [ enumproc =
                   <enumproc type> ; ] |
        codeset = ( <enum code> { , <enum code> } ) ;
      <enumproc type> ::= <procedure id>
      <procedure id> ::= <id>
      <enum code> ::= <string>

<ordered list declaration> ::=
  #orderlist : <ordered list type>
    [ name = <ordered list name> ; ]
    codetype = <code type> ;
    [ begin = <OHeader> ; ]
    [ end = <OTailer> ; ]
  #end

    <ordered list type> ::= <unsigned integer>
    <ordeedr list name> ::= <string>
```

```
      <OHeader> ::= <string>
      <OTailer> ::= <string>

<attribute frame declaration> ::=
  #attrframe : <frame type>
    <field statement> { <field statement> }
  #end

   <frame type> ::= <unsigned integer>
   <field statement> ::= <field no> <field token> { <field token> } ;
   <field no> ::= <unsigned integer>
   <field token> ::= <constant> ! [*] <slot> ! r <reserved length>
   <constant> ::= <string>
   <slot> ::= / <id> /
   <reserved length> ::= <unsigned integer>

<attribute pattern declaration> ::=
  #attrpatt : <pattern type>
     [ succ = <pattern type> ; ]
     frame = <frame type> ;
     pattern = <pattern token> { <pattern token> } ;
  #end

   <pattern type> ::= <unsigned integer>
   <pattern token> ::=
      [*] <matching string> ! <slot> ! * ( <slot> [*] <maching string>
      ( <begin pattern> ... <end pattern> )

   <begin pattern> ::= <matching string>
   <end pattern> ::= <matching string>
   <matching string> ::= <string>
```

```
<typesetting declaration> ::=
  #typesetting
    defaultsize = <size type> ;
    defaultfont = <font type> ;
  #end

  <size type> ::= <alphabet>
  <font type> ::= <alphabet>
```

3. Procedure

```
<level structure> ::=
   [<page>]@NODE<level no>,<element type>[<node descriptor>]
         [@ENDNODE<level no>]

<level no> ::= <unsigned integer>

<node descriptor> ::=
   [<header>]@DEL<text_markup1> ;
   [,<ordered list descriptor>][<header>]<nested level structure>
      {<nested level strucure>} ;
   ,T(<row size>,<col size>[,<wize>])[<header>]@DEL
      <table entry>{<table entry>}

<ordered list descriptor> ::=
   O<ordered list type> ; OS<ordered list type>{,<ordered list
            type>} ;
   OE<ordered list type>

<nested level structure> ::= <level structure>
<row size> ::= <unsigned integer>
```

```
<col size> ::= <unsigned integer>

<wise> ::= r | c

<table entry> ::=

   {[<tag>]<table entry sentence>@ETES}[<tag>]<table entry sentence>@ETE |

   @TEC | @TEA

<tag> ::= <constant>;

<table entry sentence> ::= <level structure>

<header> ::= @HEADER[,v]@DEL<header content>@ENDHEADER

<header content> ::=

   [<text_markup2>]<header name><text_markup2> |

   <text_markup2><header name>[<text_markup2>] | <text_mark2>

<header name> ::= @HEADERNAME<text_markup3>@ENDHEADERNAME
```

4. Reserved word

#concrete	#orderchar	#ordercode	
#orderlist	#textelement	#attrframe	
#attrpatt	#typesetting	#end	
charset	chartype	enumproc	codeset
name	codetype	begin	end
succ	frame	defaultsize	defaultfont