Title:	An Automated Chinese Text Processing System (ACCESS) - User-friendly Interface and Feature Enhancement
Department:	Information Engineering
Name:	Suen Tow Sunny (孫弢)
UID:	93057560
Supervisor:	Dr. Wong Wing-Shing (黃永成教授)
Date:	June 30, 1994
Course:	MSc in Information Engineering (Year 1 of 1)

Acknowledgment

I would like to express my deepest gratitude to Dr. Wing-shing Wong (黃永成), the supervisor of this thesis, for his kind help and support in the whole development process.

I would like to thank Prof. Chang-ning Huang (黃昌寧), Dr. Yang Gu (顧陽) and Dr. Wai-yin Ng (吳偉賢) for their precious time to grade for the thesis.

I should spend a few words to thank Mr. Yee-tak Loo, Mr. Kong-ngai Chung, Miss Serena Cheng, Mr. Siu-wing Cheung and Mr. Chi-nang Tang for his kind help on inputting text samples. Special thanks to Miss Oi-yi Mak for her help on document preparation, and all those colleagues and friends for providing vital comments on the thesis.

Abstract

An **A**utomatic **C**hinese text pro**CESS**ing system (ACCESS) is a system with the fundamental function of tagging Chinese text lexically to make text processing feasible with its syntactic meaning.

It is a cooperative project involving many researchers and this report documents the progress on the user-interface and some feature enhancement of the system.

The first part of this thesis is subtitled "ACCESS with an Extendible User-friendly Interface". A user interface will be built on the X window system and it aims at providing user-friendly screens for easy manipulation. In addition, the design of screen components is intended to be flexible to enable further extension on the application modules since the underlying project is targeted to be a multi-purpose system to cope with all generally-used Chinese text processing.

The second part, "Study on Algorithms for Automatically Correcting Characters in Chinese Cangjie-typed Text", will present a study on automatically correcting Chinese text input with a proposed algorithm as a feature enhancement to the text processing system.

Table of Contents

Automated Chinese text proCESSing system (ACCESS) - User-friendly Interface and Feature Enhancement

Introduction

Automated Chinese text proCESSing system (ACCESS) aims at providing a syntactic meaning to original Chinese text for further processing. At this moment, Chinese text computing is commonly limited to word processing capabilities in the commercial world. Further processing like text-to-speech synthesis or automatic language translation are not mature enough to be commercially successful due to the lack of a theoretical basis.

ACCESS generates syntactic information by providing part-of-speech tagging, which is the kernel function of the whole system. In the current implementation, an article will be entered into the system and processed with the segmentation and tagging process, with some kind of optimization algorithms, like branch-and-bound or shortest-path. The input article will then be outputted in a tagged format. Each sentence will be segmented and assigned with a part-of-speech pattern type, such as noun, verb, adjective, etc. The tagged file will be reserved in the system for further processing.

To facilitate the research on both part-of-speech techniques and future application development, an all-purpose multi-application environment is developed. This environment includes some user-friendly interfaces for user initiation since Chinese system has some technical problems in itself like input method which will create user frustration. The environment is also required to be highly extendible to include future application with high maintainability.

There are many requirements for developing an all-purpose system. Source code aims at portability. Implementation will be put on multi-platforms with multi-vendor machines. Future extension should be easily configured without deep knowledge on the original source code of the system. User-friendly interface is a must for all levels of users and graphical application look should be considered for usage initiation.

There are many more requirements for developing an all-purpose Chinese system, like incorporating different Chinese input methods, as well as handling mixed two-byte Chinese and one-byte English string with separate font type.

ACCESS is implemented with the X window system to enable multi-platform portability. It applies the X/Chinese toolkit to make Chinese computing techniques transparent to application writers, and uses source code control system to maintain a safety and convenience for group development by controlling the versions and deltas of the source programs.

For easy extendibility, each new application is only required to be accompanied with an external specification file. New application will be invoked by a menu button on the menu bar of the main window. The definition contains the menu bar or menu option label strings, and the functions to be triggered if users have chosen the corresponding options.

This user interface bases on keyboard as a text input media since other techniques, like voice input or optical input, still suffer from their unsolved technical problems or low input efficiency. But not all users are familiar with the Chinese keyboard input, and there are always typos or errors in the input text. The ill-input text will surely hinder the tagging procedures and greatly deteriorate the result. A spelling checker can ameliorate the problem in the input interface providing automatic error correction. In short, there is a great desire on a true feature-enhanced user-friendly interface.

We can trace the origins of computer techniques for automatic spelling correction and automatic text recognition for English Application back to the 1960s, and the relevant researches have continued up to this date. By contrast, few studies have been devoted to this topic in the Chinese environment, but evolving communication technology and text processing requirement, like text-to-speech synthesis, urge for error correction capabilities for Chinese text. For instance, in ACCESS, text parsing plays an important role as a basis for further processing, so typo errors may lead to wrong parsing by wrong part-of-speech composed.

We can separate spelling checking into two stages: error detection and error correction. Common English spelling checkers target on non-word errors, which are not found in Chinese applications since non-word (or non-character in Chinese) combination will cause system warning to users, and the Chinese character will be

entered once again. We focus on the isolated word correction techniques and context-based detection approaches instead.

The English spelling correction techniques provide good paths of migration to the Chinese context. On reviewing of the Chinese keyboard input history, we notice the trends on applying the standard English keyboard in the Chinese environment. In most Chinese keyboard schemes, radix input proves its efficiency and accuracy among the others. Cangjie is one of the de facto scheme proved from its consistency.

Chapter 2 of this thesis briefly reviews the Chinese keyboard input techniques and the existing English text correction researches. It is then followed by the author's research on error patterns on Cangjie input method. Similarities and differences between English and Chinese typed texts will be discussed next, and finally a proposed algorithm for automatic Chinese text correction will be illustrated.

In the proposed algorithm, real-word error techniques will be implemented. For Chinese spelling error detection, context-dependent approaches, like expectation based techniques, will be adopted to detect ill-input characters, and then similar key techniques will be used to figure out the corresponding correct characters. It is shown that future studies on natural language processing and neural network techniques will provide a substantial help for exploring new area in this field.

1. ACCESS with an Extendible User-friendly X/Chinese Interface

1.1. System requirement

By its name, ACCESS serves as an all-purpose Chinese text processing system, which could be run on multi-platforms in different vendor machines with a friendly and extendible environment for future applications. After examining various types of Chinese system and applications, we have gathered the following requirements for refinement:

1.1.1. User interface issue

a) Interactive

Although the system is not targeted on the real-time basis, the user will become frustrated on long delays. After choosing a specific function, a user expects a certain level of response delay before the result is generated. Moreover, operations which need further information from users are expected to be interactively entered step by step instead of in a batch mode. Parameters input via command-line options can be viewed as an alternative way for users' preference, but not the only way.

b) Manifest to users

There may be many functions included in a system and those functions are expected to be classified into groups and can be invoked in a logically hierarchical way. The invoking method and the usage of those functions should be initiative and manifest to users.

c) System logging

ACCESS has many computation-intensive operations, like part-of-speech tagging, and many system messages will be drawn for users' information. The size of the messages can be bulky and exceed the normal screen size, hence, they should be logged and can be reviewed by users for reference at any time.

d) On-line help

For a standard editor / viewer, there exists many word-processing functions which are to be invoked by users through the keyboard. However, those functions cannot be reflected on the keyboard. Thus it needs some memorization by the users to be familiarized with those so-called function keys. It may be an obstacle for beginners and create user frustration. On-line help, as an assistant or a tutor, will solve this problem, because it can be popped out by users at any time to reduce their burden.

e) User configurable environment

There exists personal difference among users on different aspects, like colors on visual aspect, or some user-specific default settings such as screen size. These should be separated from the system itself and edited by users without recompilation of the source programs.

f) Input method issues

ACCESS is a Chinese text processing system so text input is an inevitable issue. At this moment, the most common media to input Chinese character is keyboard input which should be adaptive to various users' need, both novices and experts. For novices, they may switch among different input methods if they cannot get the corresponding Chinese characters in one input methods. They also expect more screen response to confirm their input than experts. As there are numerous existing and new input methods, the system should allow the users to find a convenient way to add the input method without modifying the source program.

1.1.2. Development issue

a) Modularization

The system is aimed to incorporate many existing or future applications, so it should be written in a modular way for clear job division and retain high maintainability. Modularization also facilitates system profiling for fine-tuning the performance of the whole system by individual part monitoring.

b) Extendibility

There should be a clear and formally specified way to add new modules into the system, as well as an easy-to-follow method to trace back each module for modification or deletion.

c) Version control

The system may involve many programmers in the development process in the light of multiple applications. There should be a complete logging mechanism to maintain the version control on the source, recording the amendment on source code and the name of the person who applies the amendment with the modified time. There should also be some descriptions to let the others know the purpose of the amendment.

d) Portability

The system is a multi-application system for all-purpose Chinese text processing, hence, it will be implemented on different platforms in different vendor machines. The source code should be developed so that no or little modification is needed to run it on popular computing platforms.

1.2. Development decision

After detailed investigation into the above stated requirement and contemporary programming techniques, the following decision has been made for consistent application development.

1.2.1. X window system

X window system, or X, is a network-transparent window system. It is previously developed in MIT and now administrated by the X Consortium which comprises many large hardware and software vendors. It means that we can use X-applications on different platforms or machines in the network without having to rewrite, recompile or relink them. This implements the idea of being device-independent, and relieves many programmers' headaches.

The graphical user interface (GUI) environment enables users to operate the system with WIMP interfaces (Windows, Icons, Menus and Pointing devices). WIMP interfaces utilize "direct manipulation". Shneiderman (1983) [32] summarized WIMP interfaces with the following four features:

- Continuous representation of the object of interest. ("WIMP-talk" refers to things that appear on the screen as "objects".)

- Physical actions (movement and selection by using the mouse) instead of complex syntax.

- Rapid, incremental, reversible operations whose impact on the object of interest is immediately visible.

- Layered approach to learning that permits usage with minimal knowledge. Users are encouraged to "expand their knowledge gracefully".

In X window system, user configurable items are treated as resources which are separate from the application program and can be stored in a file with normal plain text representation for permanent and handy editing without recompilation of source code.

1.2.2. X/Chinese toolkit

X has an underlying principle that all the function libraries it supplies must avoid giving a specific "look and feel" to the users. This task is left for the programmer to handle. The implication of this principle is controversial. On the one hand, it may enrich the system's flexibility. On the other hand, it may be too primitive to general application programmer.

A toolkit program deals with higher-level abstractions called widgets. A widget is a user interface component like a menu, a scrollbar, a text entry field, a label, or a pushbutton. X-application programmers can then take all these existing widgets and put them together into a user interface. New widgets are also allowed to build upon the existing ones.

X/Chinese toolkit is built by Suen (1993) [34] to provide widgets on Chinese computing including both input and output facilities. On the output side, it

manipulates both two-byte Chinese text and one-byte English text with separable font type. For Chinese keyboard input, it inherits *cxterm* (from M. C. Pong and Y. G. Zhang [28]) to provide efficient and flexible hanzi input handling. It supports multiple hanzi input methods. It allows users to switch among them, and allows users to define and add any input method without modifying the source code. In addition, X/Chinese toolkit put the inputting information in a separate window to allow user to fine-tune the input method window position and input the text with various input method simultaneously for novices.

1.2.3. C language

C is a common programming language which exploits the features of simple expressions, modularized procedures and flexible data structure. It is designed for general-purpose use and hence has little constraints to provide very high portability with great efficiency.

1.2.4. Source code control system

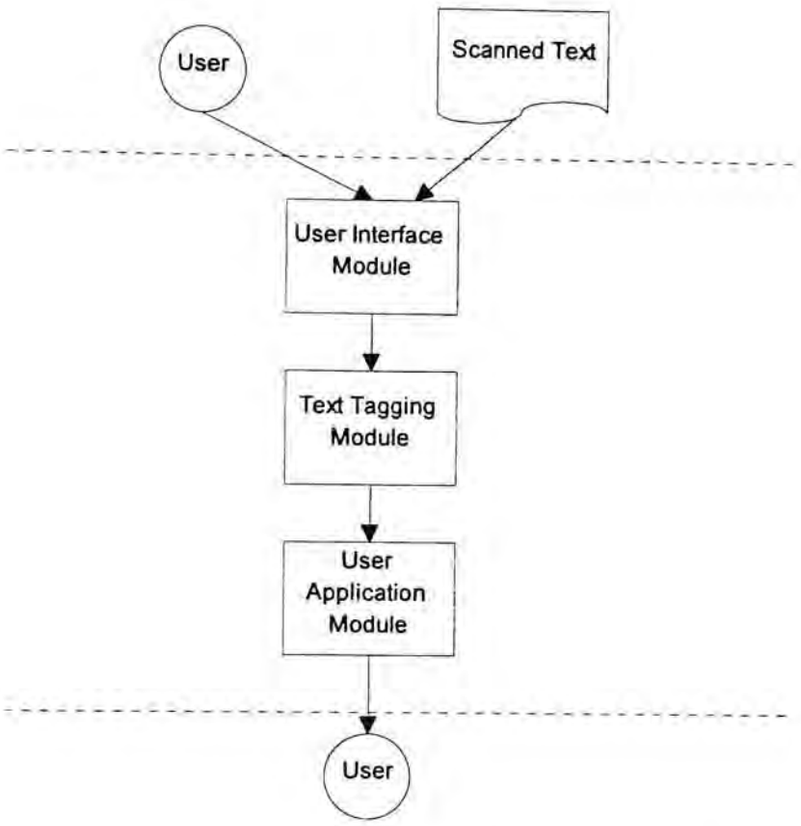
The Source Code Control System (SCCS) stores C-language code, documents, shells, or anything consisting of text, and provides version control for virtually all UNIX text files. SCCS relieves the user of the effort of maintaining versions of a file manually. It also encourages quality control and security by the use of its many and varied features. When developing C-language programs, programmers can store early working versions in SCCS, so that those versions can be recovered later if required. When changing a program, one should get the source code out of SCCS, change it, store it back, and then build the program from the SCCS source.

SCCS controls the growth of a program or document in a way that matches the maintenance of computer software: the initial development calls for incremental testing versions, an initial release, enhancements in future releases, and software fixes to existing releases. All those releases are logged with programmer-applied descriptions and modified date and time for tracing.

1.3. System architecture

The system are now developed and implemented upon the UNIX operating system and it uses X-window as its user-application interface.

The whole system is now developed in a module-based format, including 3 modules:



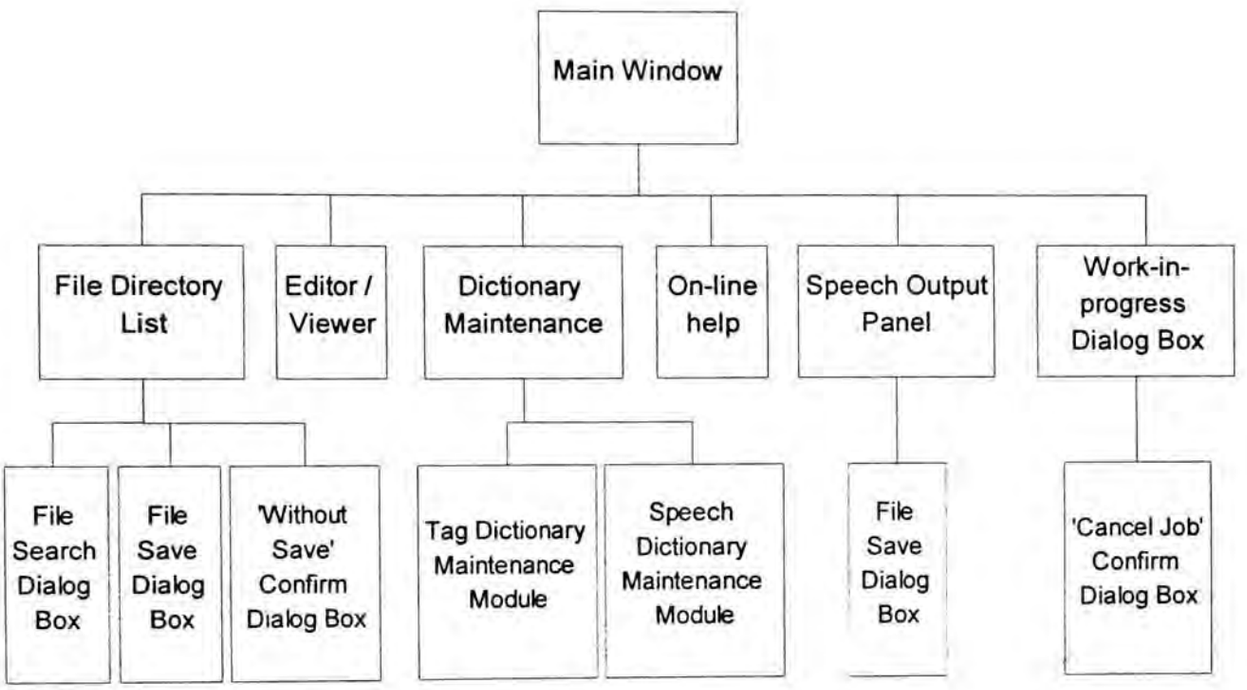
- ***User interface module***: it is composed of a Chinese environment with a Chinese text editor to let user input and edit Chinese text interactively. The user interface will be built with X-window system, and it supports windowing and mouse manipulation facilities, thereby providing a user-friendly and easy-to-use interface.

- ***Text tagging module***: this part includes word segmentation and text tagging. It is a process of giving lexical meaning to the original plain text.

- ***User application module***: it is a practical phase in the overall system. After acquiring the semantic meaning from the text tagging module, the tagged text can act as input to a whole bunch of text processing application. These applications are embedded into the system as modules to provide flexibility.

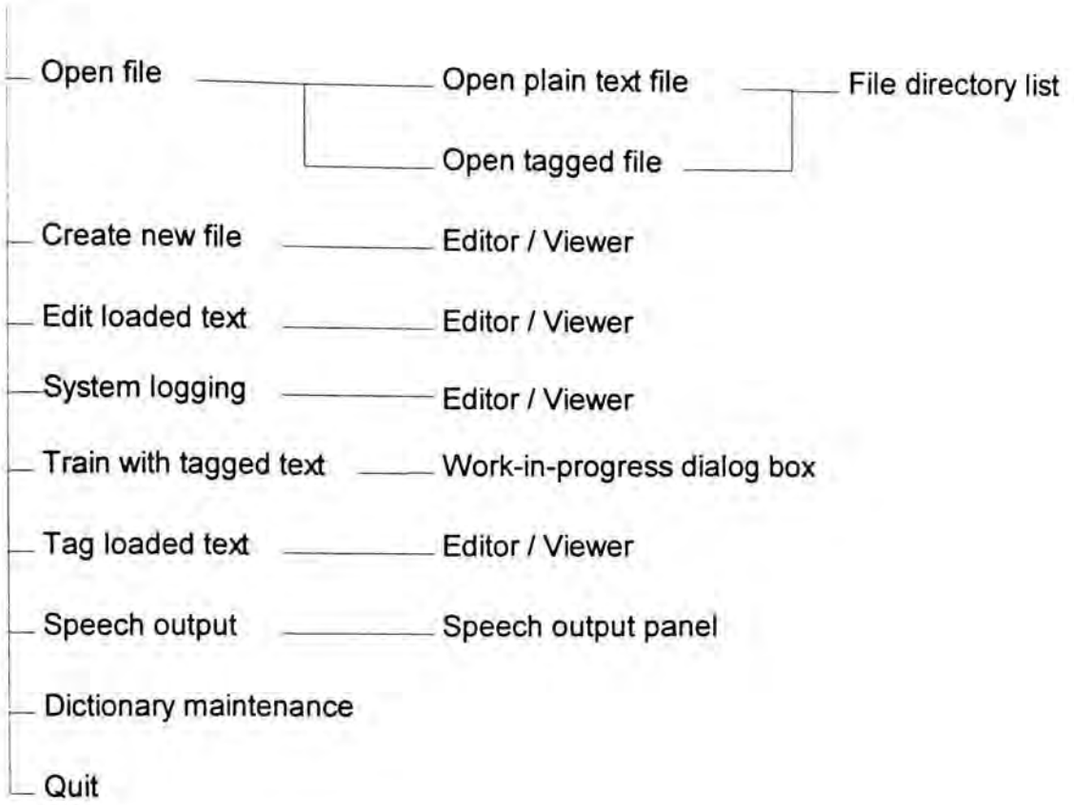
1.4. User interface

When the system is invoked, it is started with a main window to be operated as a console for the user to manipulate all functions (modules) of the system. The structure of the user-interface screen is shown below.

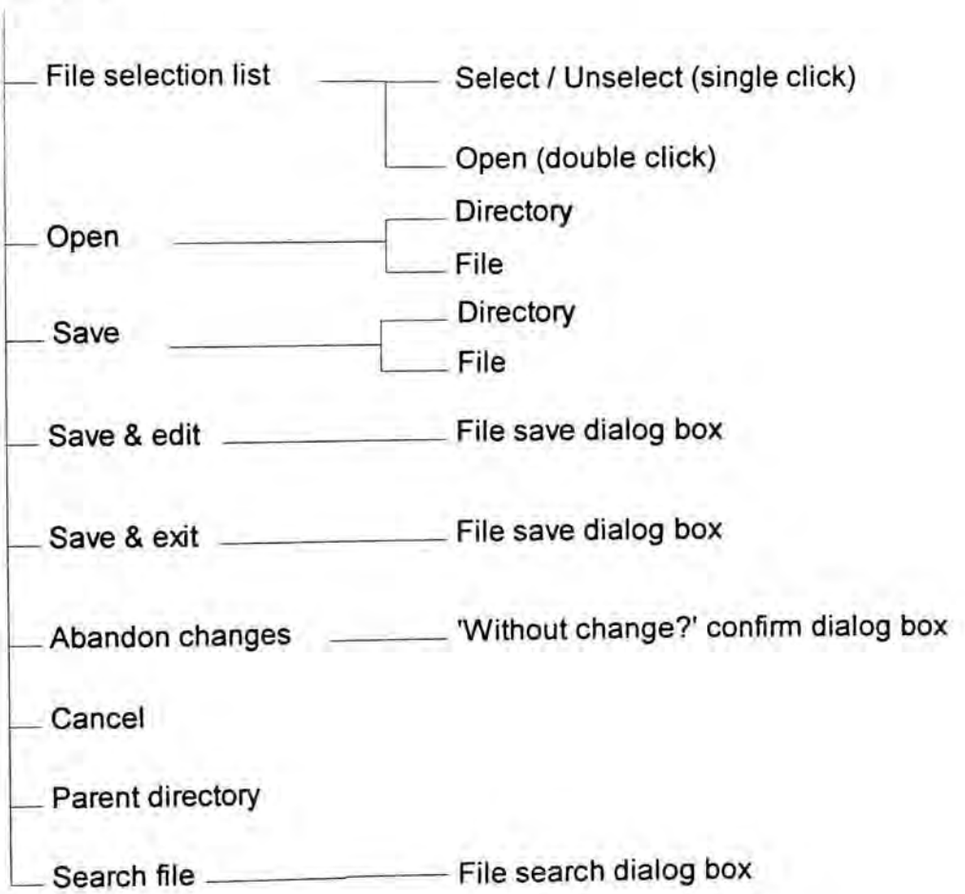


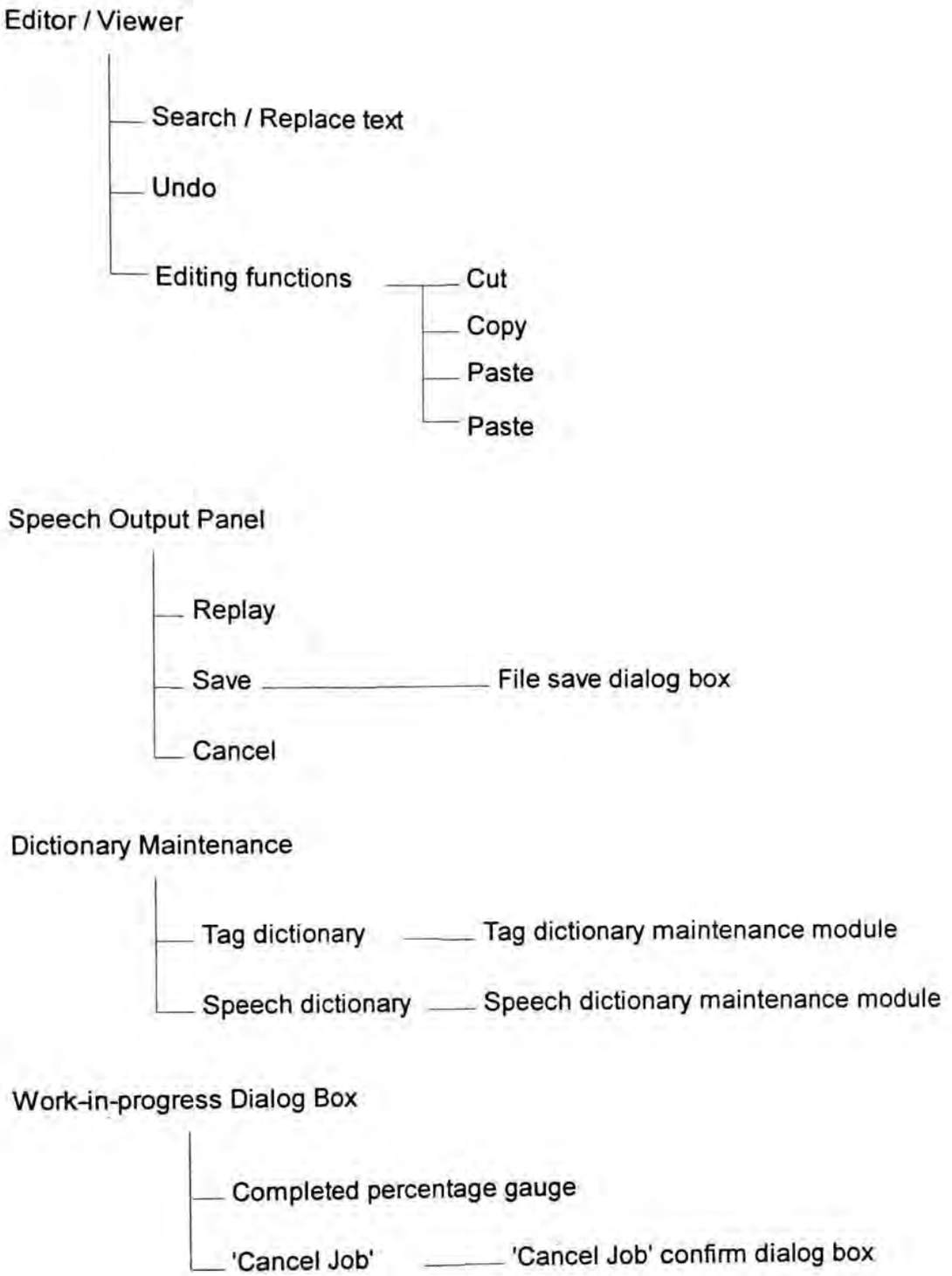
Each box above will be interpreted as a user screen, or in other words, a window frame in the windowing environment. The components on each window can be listed by the following structure.

Main Window



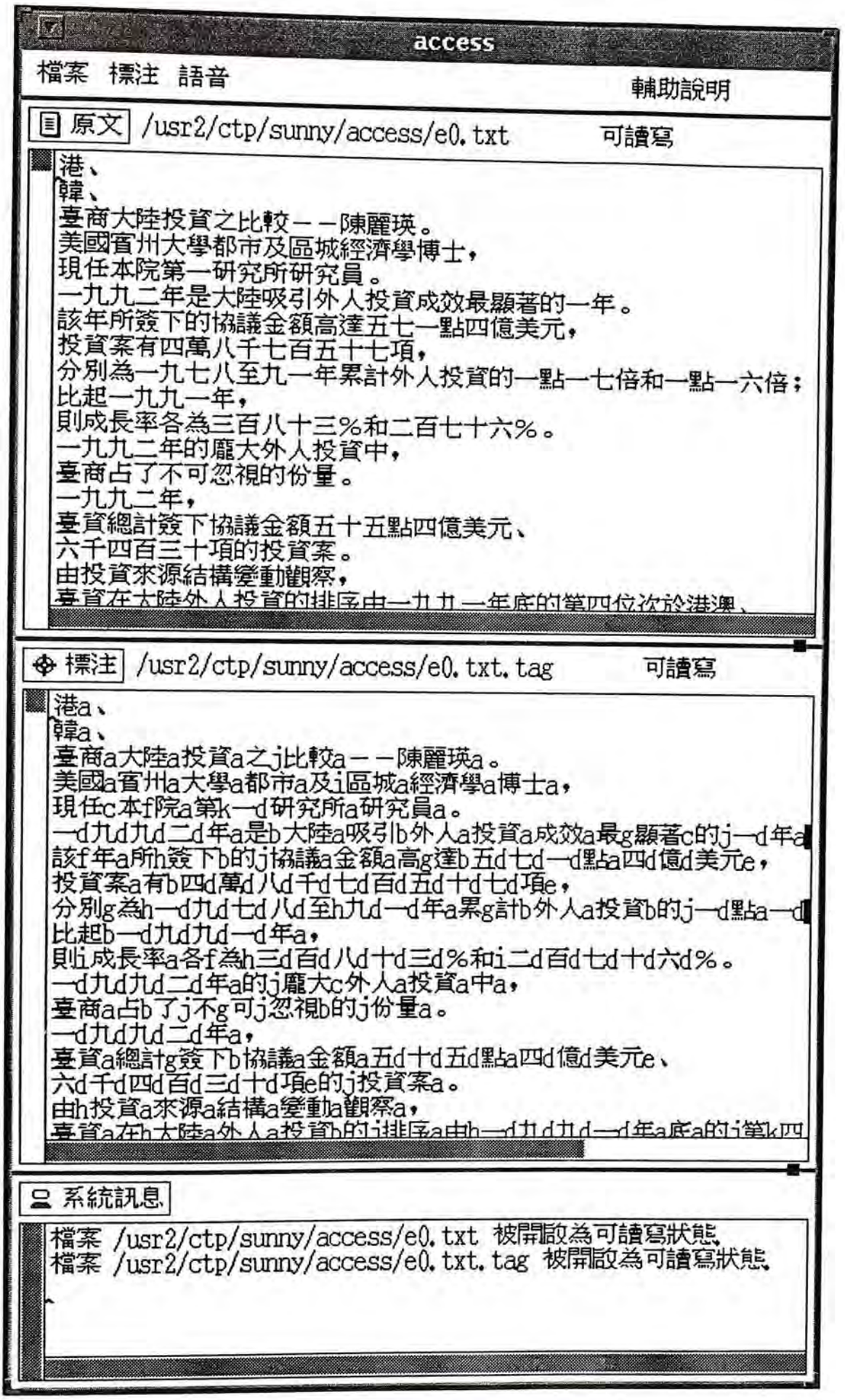
File Directory List



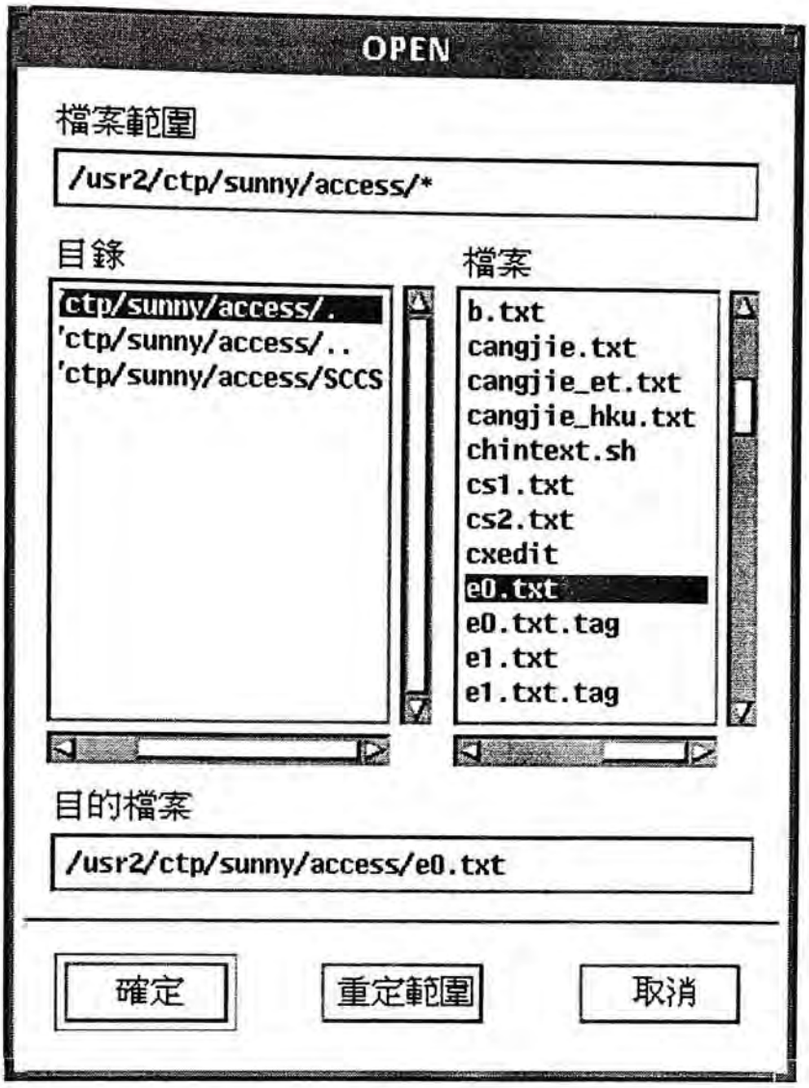

Editor / Viewer
Search / Replace text
Undo
Editing functions
Cut
Copy
Paste
Paste
Speech Output Panel
Replay
Save
File save dialog box
Cancel
Dictionary Maintenance
Tag dictionary
Tag dictionary maintenance module
Speech dictionary
Speech dictionary maintenance module
Work-in-progress Dialog Box
Completed percentage gauge
'Cancel Job'
'Cancel Job' confirm dialog box

1.5. Sample screen

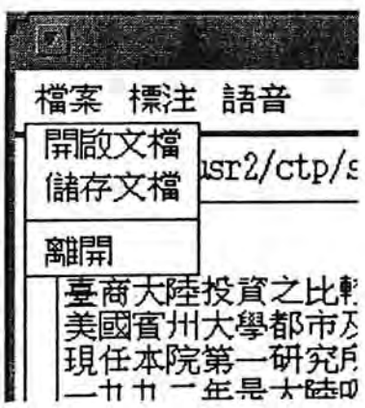
Main window



File open / save dialog



Menu options



1.6. System extension

ACCESS is an all-purpose multi-application system, which allows program extension by adding new applications. Programmers are required to follow the instructions below to extend the system.

Each new application will be represented as a menu button on the menu bar of the main window. A definition file with the filename sufficed as ".def" has to be created to declare the required information. Then the program can be recompiled once again by issuing the command "make again". The system will generate the header file "menu_config.h" once again for compilation.

For instance, suppose a new function named as "翻譯" (translate) is put to the system and there are two options, one is "中譯英" (Chinese to English) and the other is "英譯中" (English to Chinese), with a line separator displayed between those two options. The two functions to be invoked are TranslateCtoE( ) and TranslateEtoC( ) respectively.

To declare the above extension to the system, a definition file is created with the file named "translate.def":

```
#
# Module Definition
#
StartModuleDefinition
ModuleName:    Translate
MenuButtonLabel:    翻譯

#
# Menu Item #1
#
StartMenuItem
MenuItemType:  String
MenuItemLabel: 中譯英
MenuItemFunction:   TranslateCtoE
EndMenuItem

#
# Menu Item #2
#
StartMenuItem
MenuItemType:  Line
EndMenuItem

#
# Menu Item #3
#
StartMenuItem
MenuItemType:  String
MenuItemLabel: 英譯中
MenuItemFunction:   TranslateEtoC
EndMenuItem

# End of Module Definition
EndModuleDefinition
```

Term description:

(Note: the following names are case-insensitive. In other words, the capitalization of the term names is not important and to be ignored during parsing)

#: Comment prefix. The system will ignore all the things after the "#" mark.

EndMenuItem: It should always be the last statement of each menu item definition section.

EndModuleDefinition: It should always be the last statement of the file. Anything below this line will be ignored.

MenuButtonLabel: It will be shown on the menu bar of the main window.

MenuItemFunction: The callback function to be invoked if user chooses this menu item.

MenuItemLabel: It is required to be filled if its **MenuItemType** is set to **String**.

MenuItemType: It can be filled with one of the following options

Blank = blank separator
Line = line separator
String = menu item label

ModuleName: Name for the module. ('Module' means the new application group to be added to the system). To be displayed in the comment of the generated source code only.

StartMenuItem: It should always be the first statement of each menu item definition section.

StartModuleDefinition: It should always be the first statement (besides the comments) of the file.

1.7. System portability

The system is developed upon UNIX operating system with X window interface, and the whole system is coded with C language. In other words, it is produced under a high level portability environment.

However, the following part of source codes are subject to careful examination before the system is ported to other platforms:

- ***Hardware-dependent Code***: user application modules are sometimes forced to be re-written, like text speech, for instance. The digitized format and hardware operation are quite different on various platforms. This part needs to be further re-written for implementation on, say, PC.

- ***Low Level Library Function***: ANSI C serves as the standard for programs on writing programs with this language. However, the C compilers on many UNIX machines do not conform to the ANSI standard. Programmer should consult the manual before coding.

- ***External Resource File***: in writing X programs, the windowing properties, like font and border color, can be defined in the program or in a resource file. In-program resource may lead to better control of these property by programmer, whereas it also leads to decreasing of platform portability. In the project, the user-interface window properties are all defined as external resources.

2. Study on Algorithms for Automatically Correcting Characters in Chinese Cangjie-typed Text

2.1. Chinese character input

Chinese character input signifies the process to enter Chinese characters to be processed in the computer. It has been considered a bottleneck of Chinese computing.

ACCESS is designed like other standard Chinese applications to run with Chinese text. Ill-input texts greatly deteriorate the tagging results and usability of subsequent applications. Hence, in designing the Chinese input interface environment, we have to consider automatic correction on input text as a feature enhancement.

As we all know, Chinese characters, voices, images are all analog signals whereas data processed in computer are digital signals. There are mainly two ways to transform analog signals into computer: natural recognition or pattern recognition, and manual input. The former implies automatic recognition of analog signals and conversion to data signals by machines. The latter depends on human to transform text signals to data (via keyboard scan value). Obviously keyboard input is not suitable for voice or complicated image input.

Natural character recognition proves to be a convenient, optimal and advanced input method. It uses artificial intelligence to recognize and classify analog signals, and to input any complicated pattern, image, text or voice to machines. It does not require any prior training before users can input Chinese characters. We use optical character recognition reader to input Chinese text by natural recognition. It applies optical scanning device to read Chinese characters into digital signals. The signals will then pass through the process of feature extraction and comparison with recognition dictionary to produce result.

According to the recognizing object and recognition method, we can divide automatic Chinese recognition into three types: printed text recognition, hand-written text batch recognition and hand-written text real-time recognition. The former two methods save a lot of time wasted on keyboard input, and the latter one offers a simple path for beginner to approach Chinese computing environment.

Although natural recognition is an efficient way to input Chinese characters, there are still some key technical problems which are still left to be solved. and it involves high investment on hardware equipment.

Hence, at this moment, we still have to choose keyboard as a default device to input Chinese characters. Chinese characters have complicated and inconsistent structure, and so it is not easy to be decomposed. Moreover, the number of Chinese characters is also quite large. In spite of these issues, Chinese keyboard input is still relatively common in Chinese computing for its low cost and flexibility.

2.1.1. Chinese keyboards

There are 3 existing types of Chinese keyboard.

- whole character keyboard (整字鍵盤)

  It is the earliest type of Chinese keyboard. It resembles the Chinese typewriters that it chooses the Chinese character directly from the keyboard. The keyboard is set to 3000 Chinese character approximately and about 200 symbols with a few function keys. We sometimes call it "Big keyboard" (大鍵盤) because it contains quite a number of keys and the area of the keyboard is quite large in size. Recently, the keyboard is developed upon pen-based touch key to reduce typists' efforts.

- tailored keyboard (專用鍵盤)

  It uses relatively dependent Chinese character parts (radices) to assemble Chinese characters. It is similar to standard keyboard, but it contains more keys and the key arrangement is more manifest to users. In addition, the character composition rules are easier than standard keyboard. It avoids the big size and the limitation on number of Chinese character in whole character keyboard. We sometimes call it "Medium keyboard" (中鍵盤) according to its corresponding size relative to whole character keyboard and standard keyboard.

- standard keyboard (標準鍵盤)

It is the most wide-spread Chinese keyboard in current Chinese computing environment. This kind of keyboard is sometimes called alpha-numeric keyboard or "QWERTY" keyboard. It contains about 100 keys with a standard key arrangement. Since most computer has been invested on these standard keyboards, it inherits the advantage of low cost. It has helped to make Chinese system popular.

To input Chinese characters with the relatively small number of keys in standard keyboard, we have to use two or more keystrokes to compose a character. It implies that users have to learn and use the system regularly in order to be familiar with the input method. This is the main bottleneck of implementing Chinese keyboard input with the standard keyboard.

2.1.2. Keyboard redefinition scheme

At the moment, a few hundreds of standard keyboard input schemes have been proposed. We can classify these schemes into four main gropus according to the way they decompose a Chinese character into attributes.

i) Radix input

Radix input decomposes the Chinese character glyph by preset rules and provides some sorting and coding techniques on those radices (字根). There are numerous radix input schemes which we can further divide into stroke coding (筆劃編碼) and part coding (部件編碼).

Stroke coding assigns numbers or English letters to a small number of basic strokes. When typists input the strokes in a certain sequence, the corresponding Chinese character can then be mapped out.

Part coding contains much more basic elements (more than 100 in some particular schemes). Like stroke coding, the input method records typists' input sequence of those parts and generates a Chinese character according to the sequence.

The "sequence" here mostly means the writing sequence of the Chinese character, so it should pose no problems to native writers. Since the number of parts may exceed 100, each key in the standard keyboard will be assigned to a few Chinese character parts.

Comparing these two coding approaches, stroke coding is superior for its ease of use. It needs only a few keys to represent all the basic parts to reduce the memorizing jobs for beginners. However, it suffers from longer input code due to its low coverage for each stroke relative to the whole Chinese characters.

On the contrary, part coding is more complicated but needs less radices (no more than 5 radices) for each character. Its advantage is obviously the efficiency in inputting lots of Chinese text, so it becomes the main stream of Chinese keyboard input although there exists a certain level of difficulty for beginners.

In summary, radix input is most suitable in the static input environment which means to type Chinese with a written or printed Chinese script. Typists needs only to look and type directly.

ii) Phonetic code input

This kind of input schemes requires typists to input basic phonetic symbols according to the pronunciation of the Chinese character. In contrast to radix input, it is more familiar to most native Chinese speakers. It can be further divided into phonetic word input (羅馬拼音) and phonetic symbol input (音標拼音).

Phonetic word input based on the "Chinese Pinyin Scheme" <<漢語拼音方案>> uses the 26 alphabet letters to input the Chinese character. The code length is variable, ranging from 1 to 6. For instance, to input the character 漢 (pronounced as "Han"), we have to type the 3 English letters "H", "A", "N" to generate the Chinese character. Like stroke coding, it suffers the weakness of low efficiency, but it is easy to learn for the people who have already been familiar with standard English keyboard typing.

Phonetic symbol input will improve the efficiency with less keystrokes and constant code length. It assigns the phonetic vowels (韻母), consonants (聲母) and tones (調) to standard keys and typists can use one key for each of the vowels, consonants and tone combinations to input a Chinese character.

In contrast with radix input, phonetic code input schemes are suitable for typists without typing script. In other words, it performs better in the "think and type" or "listen and type" environment. It also avoids the influence of different font type which hinders the typists' comprehension.

The above two input schemes suffer from a fatal problem in repeated code (重碼). In the theory of information coding, the basic requirement always stresses on one to one mapping. However in Chinese input coding, we have to give up this requirement by reducing the code length for the sake of efficiency. Thus, the problem of repeated code is inevitable and it leads to inaccuracy and confusion in the typing process. Especially in phonetic code input, even though we have the vowel-consonant-tone combination, characters with the same combinations can still be numerous. The following two input schemes are sometimes adopted for the sake of unique identifiability.

iii) Combination of radix, phonetic and meaning

The problem of repeated code arises due to a many-to-one mapping between characters and codewords. This group of input schemes directly resolve the problem by extending the codeword based on radix and phonetic to include character meaning.

For instance, 己 (self) and 已 (already) are repeated code with the same pattern in radix input but we can append their consonants for further classification, i.e., "g" and "y" respectively (In Cantonese, 己 pronounces as "gei2" and 已 pronounces as "yi5"). There are various schemes in implementing these kinds of combination and the above is just one example.

This combined input scheme is always appreciated on its efficiency and accuracy to avoid choosing the alternative repeated code from the screen after entering the input code, since the rate of repeated code should be relatively low. Nevertheless, its popularity is still a question due to its complicated structure and the extra requirements on memorization.

iv) Serial code input

It means to arrange the Chinese characters in a certain regular sequence and those ordered Chinese characters will be assigned with a unique code according to its position. Typists are required to key in the code accurately to get the corresponding Chinese character. There exists two types of serial code input: telecode (電碼) and internal code (內碼).

Telecode is commonly used in telecommunication by telegraph with a long history, so it can still be found in some Chinese terminals. It uses 4 decimal digits to represent about 7000 Chinese characters. The existing telecode standard is based on the 1981-issued "Standard telecode table" << 標准電碼本 (修訂本) >>.

There are numerous types of internal code in the history of developing Chinese computer systems. Up to now, we can temporarily conclude two main streams: National standard GB code for simplified Chinese character and *de facto* standard BIG-5 code for traditional Chinese characters. There is a new trend for the Unicode to achieve global standard of internal codes but we have to wait, to see whether it will become popular.

Serial code input can totally avoid repeated code. Machine implementation will be straight forward and low computation is required. After a typist memorizes the internal code, he/she can achieve optimal efficiency in Chinese keyboard input. However, we can hardly find its usage outside professional typists.

2.2. Cangjie input method

Weighing the different criterion among efficiency, accuracy, rate of repeated code and level of difficulty, we can easily understand why radix input methods are popular. In simplified Chinese character environment, the leading radix input method is Wubi input method (五筆字型輸入法), whereas in the traditional Chinese character environment we find Cangjie input method (倉頡輸入法). Wubi is proposed by Wang Yong-min (王永民) in 1983. Cangjie is proposed by Zhu Bang-fu (朱邦復) with a number of different version available in the commercial world. The latest version of Cangjie is the 5th edition (issued in 1985) but the most popular is still the wide-spread 3rd edition (first issued in 1982).

This research is proposed under the project Automated Chinese text proCESSing system (ACCESS). In the system, we started research with traditional Chinese characters, Cangjie input method was adopted and studied. Since Cangjie resembles the other input methods and the manipulation of standard keyboard, it is not difficult to extend the research result to other input methods.

Cangjie uses 24 English letters (A - W and Y) as basic radices. X key is used for complicated parts and Z key is reserved for repeated codes. In some implementation, Z will be added to the front of second Chinese input code for the case of repeated code. These 24 keys are assigned with a total of about 60 parts to compose about 40,000 Chinese characters. With the standard keyboard, each Chinese character does not exceed 5 keystrokes. From the view point of repeated code, Cangjie is quite effective in reducing the number of crashed code. A Cangjie radix table has been printed in Appendix A (page 51) for reference.

Cangjie decomposes a Chinese character into head part (字首) and body part (字身). The head part contains one to two radices and the body part contains 1 to 3 radices. Then each part will be assigned radices with the sequence of "left to right, top to bottom, and outside to inside" (由左而右, 由上而下, 由外而內) .

The implementation of Cangjie in Chinese system has not been included in the scheme and left to the decision of system developers. The following illustrates the most common implementation.

A Chinese character will be generated after the typist has entered all the corresponding radices and pressed the <space> key (or <enter> key). If a Chinese character can be successfully mapped from the table, the typist can continue to type the next character. Besides, there exists some error exception handling routines:

i) repeated code

If the system has mapped more than one Chinese characters according to the input radices, all the alternatives will be output to the screen and wait for the typist to choose (some system will produce a "beep" sound to warn the typist). The typist can scan through the list and type the corresponding number to select the correct character. Or he/she can choose to continue to type the next character. Then the system will automatically choose the first alternative for

the typist. In most cases, the system will only automatically choose for the typist if the user continues to type the alphabet character. If the typist just types punctuation key while the system waits for choosing, only the punctuation mark will be output and the system still waits for the typist. It leads to the "repeated code and punctuation error" (which is described in the section of "Research on error rates in Cangjie input method" (page 41).

Some (early) systems do not list the alternatives and directly output the first choice to the screen. If the typist wants to get the second alternative, he is required to prepend the "Z" character to the radix code. One more "Z" will be prepended if the required character is the third choice, and so on.

ii) no character mapped

There are two conditions leading "no character mapped" errors. Typists may wrongly type or compose the radices, or the size of character dictionary in the system is insufficient (like the 13094 BIG-5 standard, many ancient Chinese characters and many Hong Kong dialects have not been included in the table). The system will respond with a "beep" sound (some dedicated system will produce a "beep" sound different from the "beep" sound of the repeated code condition) and an error message. Typists are expected to input the whole sequence of radices once again.

2.3. Review on existing techniques for automatically correcting words in English text

We can trace the origins of computer techniques for automatic spelling correction and automatic text recognition back to 1960s, and it has continued up to now. In the commercial world, we can see that there are many popular English spelling checkers, but those programs are limited to their scope and accuracy. They are mostly focusing on isolated words, with taking into account any information that might be gleaned from the linguistic or textual context in which the string appears. Such isolated-word correction techniques are unable to detect the significant portion of errors, including typographic, phonetic, cognitive, and grammatical errors, that result in other valid words. There is still a long distance to achieve an optimal automatic correction.

Before proceeding to the existing techniques in spelling correction, we have to distinguish between the two tasks of spelling correction: error detection and error

correction. We first have to detect the ill-input strings (such as through matching with the dictionary or lexicon) and then correcting the misspelled string. The latter task will prove to be a much harder problem than the former.

The following literature review on automatic English word correction is mainly based on the corresponding review study by Karen Kukich (1992) [15]. According to Kukich, automatic word correction research are now viewed as focusing on three increasingly broader problems:

- nonword error detection
- isolated-word error correction
- context-dependent word correction

2.3.1. Nonword error detection

There are two main techniques on nonword error detection, which are called "n-gram analysis" and "dictionary lookup".

N-gram are n-letter subsequences of words or strings, where n is usually one, two, or three. N-gram techniques usually require either a dictionary or a large corpus of text in order to precompile an n-gram table. There are subtle problems involved in the compilation of a useful dictionary for a spelling correction application. For instance, the simplest case is a binary bigram array, which compose a two-dimensional array of size 26 × 26. Each elements in this array represents all possible two-letter combinations of the alphabet.

Nonpositional bigram analysis will separate each word (string) in the text, and calculate its bigram weight. Low-weight bigrams will be prompted out for further error correction. Positional bigram analysis adds one more criteria of bigram position in the original string. Some studies devoted on trigram analysis (Zamora et al., 1981 [39]) found that although trigram analysis was able to determine the error site within a misspelled word accurately, it did not distinguish effectively between valid words and misspellings. Another study (Morris and Cherry 1975) proposed to generate the table directly from the document to be checked. It can reduce the size of the n-gram dictionary substantially but the result may be affected by the size of the document.

Dictionary lookup is a straightforward task. Researches on this topic mostly contribute to developing fast access to a dictionary. Many existing algorithms use hash table or tries search, which are proposed by Knuth in 1973. Turba (1981) [35] criticizes that the main disadvantage of hash table is the need to devise a clever hash function that avoids collisions without requiring a huge hash table. Peterson (1980) [26] suggests partitioning a dictionary into three levels for spelling detection for efficiency. These three levels are expressed in terms of storage media. Most frequently used words will be stored in cache memory (first level), domain-specific words will be put into regular memory (second level), and other less frequently used words will be stored in secondary memory (third level).

It is suggested that dictionaries must be carefully tailored to the domain of discourse of the application in order to avoid frustrating the user with too many false acceptances and rejections.

N-gram analysis is mostly applied on text recognition systems, whereas dictionary lookup is commonly seen on spelling checkers.

2.3.2. Isolated-word error correction

Most of the existing spelling checker can achieve the above defined nonword error detection efficiently in terms of the current abundance of processing power and mass memory size. Users are accustomed to expect spelling checkers to suggest corrections for the nonwords they detect. It may not suit the increasingly focused environment -- text-to-speech synthesis, which require both error detection and correction to avoid user intervention. There comes the problem of isolated-word error correction.

Researches are carried out on three issues: lexicon issues, computer-human interface issues, and spelling error pattern issues. Lexicon issues contain the dictionary construction issues which can be reviewed under the research on the above dictionary lookup.

Computer-human interface issues are typically denied by the trade-off in balancing the need for accuracy against the need for quick response time. In other words, letting the computer automatically choose the correct word for the user should be time-saving, with a risk of wrong selection. On the

contrary, permitting users to examine each detected error and choose the correct substitutes would prove to be most accurate but it could be very frustrating to the user.

Spelling error patterns can vary greatly depending on the nature of the specific application task. For example, transcription typing errors tend to reflect typewriter keyboard adjacencies, and optical-character recognizers produces more on featural similarities. There is no need to overgeneralize findings in discussing spelling error patterns. Here we will have a closer look on the transcription typing error patterns only.

Since the nonword spelling error detectors are becoming mature in these days, the relative ratio of real-word errors to nonword errors is probably higher today.

2.3.2.1. Spelling error patterns

Spelling error patterns can be discussed in the following aspects:

- Basic error types

  Damerau (1964) [10] found that approximately 80% of all misspelled words contained a single instance of one of the following four error types: insertion, deletion, substitution, and transposition. They can be further combined into multi-error misspellings. Honestly, the figure "80%" may not be true in general but it suggests that it stands for a significant portion of spelling errors.

- Word length effects

  It is generally found that most misspellings tend to be within two characters in length of the correct spelling. Unfortunately, shorter in word length will often suggest harder to correct since it implies less intraword contextual information available. For instance, it is very difficult to detect the error arising from misspelling the word "in" to "on". Hence, to design a spelling correction, the actual characteristics of the spelling errors of the intended application should be first determined.

- First-position errors

  Fewer studies have really been devoted to research on this kind of errors as it is generally believed that the frequency of errors happening on the first position in the string is relatively low. Discounting first-position errors will drastically reduce the lexicon size and boost the response time, but care should be taken to balance the trade-off against the deteriorated accuracy.

- Keyboard effects

  Grudin (1983) [12] found that the majority of expert errors with the keyboard effects were insertions that resulting from hitting two adjacent keys simultaneously while the majority of novice errors were substitutions. He also observed that 58% of all substitution errors involved adjacent typewriter keys. There is a trend to the spelling checker to include the probabilistic tendencies that arise from keyboard adjacencies and letter frequencies.

- Heuristic rules and probabilistic tendencies

  To derive a spelling correction techniques, there are two main approaches: a rule-based spelling correction algorithm (Yannakoudakis and Fawthrop, 1983 [37]) and probabilistic tendencies, such as which letters and which positions within a word are most frequently involved in errors (Zamora, 1983 [39]). There is another study (Kernighan et al., 1990 [14]) to compile probability table separately on each of the above stated four classes of errors.

- Common misspellings

  To reduce the size of lexicon and computational complexity, lists of common misspellings can be employed in the spelling detection and correction process. Such kind of lists can be found in specific dictionaries, like Webster's New World Misspeller's Dictionary which identifies 12 classes of common errors. But very few academic researches have been carried out on this topic.

Real-word errors which cannot be detected by isolated-word error correction techniques will be left to research on context-dependent word correction techniques.

After identifying the basic error patterns, we can have a look at the existing isolated-word error correction techniques from three subproblems:

a) detection of an error
b) generation of candidate corrections
c) ranking of candidate corrections

2.3.2.2. Correction techniques

Besides the n-gram statistics and dictionary lookup which was mainly used in the early years, many other clever techniques have been invented basing on:

- minimum edit distance techniques

  Wagner (1974) [36] defines the term minimum edit distance as the minimum number of editing operations (i.e., insertions, deletions, and substitutions) required to transform one string into another. In general, minimum edit distance algorithms require m comparisons between the misspelled string and the dictionary, where m is the number of dictionary entries. Mor and Fraenkel (1982) [21] invoked a hash function to look up misspellings. Some use it as an alternative to dynamic-programming techniques for improving search time in minimum edit distance algorithms.

- Similarity key techniques

  These techniques try to decompose every string into keys and arrange the keys in a specific orders. The decomposed key sequences will be used to map the corresponding sequences in the dictionary for correction suggestions. Similarity key techniques have a speed advantage because it is not necessary to directly compare the misspelled string to every word in the dictionary. In addition, the error

position can be avoided to provide a more justified view on the error sources.

Pollock and Zamora (1984) [27] devised a similarity key technique called SPEEDCOP. They classify the alphabet into skeleton keys and omission keys, in which vowels will be rank lower than consonants to preserve the word skeletons.

- Rule-based techniques

Rule-based techniques attempt to represent knowledge of common spelling error patterns in the form of rules for transforming misspellings into valid words. These techniques rank the strings with assigning scores and invoke appropriate rules to correct the error.

- Probabilistic techniques

Two types of probabilities have been exploited: transition probabilities and confusion probabilities. The former represents probabilities that a given letter (or letter sequence) will be followed by another given letter. This kind of probabilities is language dependent. The latter are estimates of how often a given letter is mistaken or substituted for another given letter.

- Neural Net Techniques

Neural nets are recently researched for spelling corrects because of their inherent ability to do associative recall based on incomplete or noisy input. They can learn from actual spelling errors and they have the potential to adapt to the specific error patterns. Studies have been devoted to designing a single chip with neural net techniques to process spelling correction in word processing machines.

2.3.3. Context-dependent word correction research

With the above techniques, nonword error can be corrected to a certain level, but those methods never touch the remaining problem -- real-word error. For instance, "from" is mistyped as "form". This type of errors is beyond the

capacity of the above techniques. It seems to require information from the surrounding context for both detection and correction. On the other hand, it should be helpful to include contextual information for improving correction accuracy for detectable nonword errors.

Before discussing the techniques in context-based spelling correction, we first have to be aware of the difficulties of detection and correction of errors in this paradigm because:

a) lexical and grammatical coverage is necessarily incomplete; and
b) locating the source of a parsing failure is often difficult (e.g. false alarm can happen on proper nouns)

2.3.3.1. Natural language processing approach

Various approaches to dealing with ill-formed natural language input can be loosely classified into three general categories:

- Acceptance-based techniques

  The techniques assume that errors can simply be ignored as long as some meaning interpretation can be found to the given application. It tolerates the errors in linguistic information and accepts heavy use of semantic information. The approach has much psychological plausibility. At present it is impractical for processing unrestricted text because it requires a semantic model of the domain of discourse and a rich formal semantic knowledge base for unrestricted text it requires is unavailable.

- Relaxation-based techniques

  En revenge, relaxation-based techniques stress that no errors can be ignored. When a parsing failure occurs, the system attempts to locate an error by identifying a rule that might have been violated and by determining whether its relaxation might lead to a successful parse. Locating and relaxing rules allow for both detection and correction of errors.

The techniques focusing mainly on syntactic processing, are the least knowledge intensive of the three approaches and for that reason the most well suited for use on unrestricted text.

- Expectation-based techniques

Expectation-based techniques are somewhere in between of the above two approaches. They acknowledge both that errors occur and that people draw on contextual knowledge to correct them.

During the parsing stage, the system builds a list of words it expects to see in the next position based on its syntactic, semantic, and sometimes pragmatic and discourse structure knowledge. If the next term in the input string is not in the list of expected terms, the system assumes an error has been detected and attempts to correct it by choosing one of the words on its expectation list. The techniques exploit various levels of linguistic knowledge and pragmatic and discourse structure knowledge.

- Parser-based writing aid tools

Kempen and Vosse (1990) [13] have produced some encouraging results with the application of Dutch parser-based contextual spelling correction. It processes a document in four stages:

a) preprocessing: removes typesetting markup symbols and corrects punctuation
b) word-level analysis: calls upon a spelling corrector to generate a small set of candidate corrections for each word type in the document
c) sentence-level analysis: a unification-based (with relaxation rules) parser is called to produce a high-level parse
d) text resynthesis: regenerates the original document with suggested corrections and error diagnostic messages

Recent research has led to significant advances toward robust natural language parsing of unrestricted English text.

Besides natural language processing based on contextual spelling correction, there are other approaches with the statistical language modeling techniques. Statistical language models are essentially tables of conditional probability estimates for some or all words in a language that specify a word's likelihood to occur with the context of other words.

2.3.3.2. Statistical language model

The probability estimates comprising SLMs are derived from large textual corpora can be very large, so early research will seldom touch this area in terms of computational resources and processing power.

Statistical language models approach cannot be totally distinguished from the above natural language processing approaches since it shares a basic premise with expectation-based NLP approaches. The probability estimates can be based on the a priori expectations for possible choices. Thus, low-probability (or zero-probability) word sequences might be used to detect real-word errors, and high-probability word might be used to rank correction candidates.

It is still premature to talk about the result of this approach, and we can temporarily divide it into the following streams:

- Detecting real-word errors via part-of-speech (POS) bigram probabilities

  Atwell and Elliott (1987) [7] developed the CLAWS tagger system to assign syntactic-category tags based on POS bigram transition probabilities to the entire 13,500 word corpus. They hypothesized that a sequence of two low POS bigram probabilities might indicate an occurrence of an error, and they set about determining an optimal threshold setting for those probabilities such that the number of errors detected would be maximized while the number of false alarms would be minimized.

- Improving nonword correction accuracy via word bigram probabilities

A more general but local probabilistic approach to word correction relies on the use of word bigram or trigram probabilities rather than just POS bigram probabilities.

Church and Gale (1991) proposed a 3-step approaches on implementing this technique:

a) a program (called *correct*) is used to generate a set of correction candidates and their probabilities for the nonword
b) word bigram probabilities were used to estimate left and right context conditional probabilities for each candidate
c) candidates were scored and ranked by computing a simple Bayesian combination of the three probabilities for the candidate itself and its left and right context conditional probabilities.

- Neural net approach

Like the isolated-word research, studies have been carried out on exploring for context-dependent word recognition and error correction in neural net modeling. Unfortunately, neural network models are suffering from the fact that expectations are implicitly represented as weights in the nets. Thousand of nodes representing words and hundreds of thousands of weights representing strengths of association between words must be fully contained in a working memory during training and processing. Those demands make NN models impractical for vocabularies larger than a few thousand words at most. In contrast, statistical language models avoid this problem as they can store their expectations on secondary storage for comparing and revising.

The conceptual basis of neural net modeling proves to be an intelligent way on text correction in terms of its adaptability. Or maybe in the future, studies can be devoted again to neural net if cleverer neural net modularization techniques become available.

2.4. Research on error rates and patterns in Cangjie input method

To investigate the error rates in Cangjie, four distinct articles have been distributed to 5 Cangjie typists with average typing rate of 35 characters per minute. These four articles are divided into two topics -- computing and medical, to avoid the jargon bias on specific topics. The size of articles are all over 1000 characters (Computing articles: 1138 and 1740 Chinese characters, and medical articles: 1818 and 1152 Chinese characters, excluding punctuation and English characters). Appendix B (page 52) contains the full copies of articles for reference.

To unify the inputting process, the following instructions have been established for typists' reference:

INSTRUCTIONS FOR TYPISTS:

1. Please record the sequence of passages typed.
2. Please record the typing time of each passage.
3. Please specify the Chinese system you are using.
4. Please save the typed file in BIG-5 plain text format. (Please specify the internal code type you use if you are not using BIG-5)
5. Please use Cangjie instead of Simplified Cangjie.
6. If possible, type each complete passage at one time.
7. If you find some errors in the original text, just neglect the errors and type accordingly.
8. Please don't correct any typing mistake during / after typing. In other words, just leave the error and DON'T press BACKSPACE to correct even you find the character has been typed wrongly. The errors in the file are very important because we are researching on the errors.
9. You may use two-byte or one-byte punctuation mark in the text, and they will be treated as the same. If there are any punctuation marks you don't know how to type or not easy to be typed in your Chinese editor, just neglect them and type two spaces instead.
10. During typing, after you have entered the Cangjie code, if there are no Chinese characters coming out (because of wrong code), you can try to input the character again. However, if one character has come out (no matter it is right or wrong), don't try to correct it and continue to input the next character.
11. For any character you forget the Cangjie code, you may press two spaces to substitute the original character and continue to type.

A program has been written to compare the typed text with the original copies. Only difference in Chinese character has been located with the negligence of wrong punctuation and English characters. Total error rate is calculated as 1.5%. The full list of errors can be found in Appendix C (page 61).

After compiling the errors, the following types of errors are located:

(Note: The bracket shows the proportion of errors occurring to total errors)

1. Nearby key error (30%)

   One of radices has been substituted wrongly by its nearby key on the keyboard.

   [E.g. 但 (人日一) (OAM) → 們 (人日弓) (OAN)]

2. Omission error (One key omission: 22%, two key omission: 2%)

   One or more radices have been omitted while the Chinese character is being input.

   [E.g. 名 (弓戈口) (NIR) → 夕 (弓戈) (NI);
   何 (人一弓口) (OMNR) → 丘 (人一) (OM)]

3. Repeated code error (11%)

   As stated above, Cangjie input method cannot totally avoid repeated code, though the number of repeated code is relatively small in contrast to other input methods. Typists sometimes forget to choose the second repeated code if the input character is not the first alternative. When typists are not aware of the repeated code and continue typing, most Chinese system will automatically choose the first character for typists.

   [E.g. 詢 (卜口心日 <space> 2) → 詣 (卜口心日 <space> 1)]

4. Character recognition / concept error (18%)

   This group of errors can also be called human knowledge error.

First kind of errors are produced by human recognition error of Chinese characters. It may be caused by several reasons. One of the main reasons is up to the typing script. In this research, we use laser-printed articles with 12-point Mingli Chinese fonts (明體) as typing script for typists. It should be the standard font size for desktop view. However, in the real world, the typing scripts are mostly hand written, it should greatly deteriorate the result because:

i) the blurred hand-written Chinese characters may mislead the typist;
ii) there are various ways to write a Chinese character, e.g. 群 can be written as 羣, but Cangjie only include the former one. In this case, printed script will avoid this error since only 群 will be printed. However, there are seldom printed script ready for typing.

[E.g. 模 (木廿日大) (DTAK) → 摸 (手廿日大) (QTAK)]

Sometimes errors can be produced by the personal knowledge of Chinese characters.

[E.g. 卻 (but) → 郤 (surname)]

Another kind of human knowledge error is concept error. It is produced by the users' wrong Cangjie concept. You may refer the following examples to the Cangjie table in Appendix A (page 51) for reference.

[E.g.
1. 東 (木田) is misunderstood as 「木日」 which generates 杳.
2. The first radix of 順 (中中中金) is 中 (|), being misunderstood as 竹 (/), and the head part (字首) of 順 should be 「 | 」 but being misunderstood as 「川」. These two errors generate the character 頎.]

5. Sequence error (5%)

The radices of the code have been entered correctly with wrong sequence.

[E.g. 小 (弓金) (NC) → 釓 (金弓) (CN)]

6. Extra radix error (2%)

One more radix key has been pressed wrongly.

[E.g. 法 (水土戈) (EGI) → 澩 (水廿土戈) (ETGI)]

7. Repetitious radix error (3%)

It is a specific case of the above extra radix error and omission error. This kind of errors will easily be caused while the radices of a character contain repetitious radices.

[E.g. 炎 (火火) → 焱 (火火火) (extra radix error)
例 (人一弓弓) → 仃 (人一弓) (omission error)

8. Mixed errors (9%)

Those errors are composed of two or more of the above stated errors.

[E.g. 以 (女戈人) (VIO) → 戍 (戈女) (IV) (combination of omission error and sequence error)

9. Others (1%)

These kinds of errors should be counted as minor ones but listed here for reference.

a) number-alphabet error

It is a special case of nearby error. A number will be produced in front of the wrong Chinese character.

[E.g. 可 (一弓口) (MNR) → 4丁 (一弓4) (MN4)

b) space between radices

It is a special case of extra radix error but space is overtyped instead of radix to produce two Chinese characters.

[E.g. 相 (木月山 <space>) → 束山 (木 <space> 月山 <space>)

c) repeated code and punctuation

The usual way which users treat repeated code is to continue typing since normally (but not necessarily) the first choice of the repeated code is the one with highest frequency. The computer will automatically respond the first chose while users continue to type the next character. It brings some problems when the repeated code happens at the end of sentence. The punctuation key will not bring the first choice to screen as usual, which produces typographical errors.

[E.g. 員, → ,員]

2.5. Similarities and differences between Chinese and English typed text

Up to this moment, we can hardly find any commercial Chinese spelling checker. This obviously relates to the conceptual difficulties of implementing a Chinese spelling checker. Mature commercial English spelling checkers mainly deal with nonword errors but we do not find such kind of errors in Chinese text. A Chinese character will not be generated if the radices input cannot map to the corresponding Chinese character. In short, there are no such nonword error in Chinese typing environment, whereas only real-word errors exist. Even the real-word error detection and correction techniques in English application is still not mature enough, we can hardly hear about the Chinese spelling checkers in current localized Chinese applications.

To make the problems clearer and more specific to Chinese text, we should first look inside the similarities and differences on Chinese typing from English typing:

2.5.1. Similarities

Cangjie input method resembles English typing in various ways:

1) In view of contextual information, both Chinese and English sentences can be divided into part-of-speech n-grams or vocabulary n-grams.

2) Cangjie redefines the 26 English letters to its radices. In other words, they have the same basic elements.

3) In English typing, we count the typing rate by assuming that the average word length is 5 letters. Cangjie also limits its radix code length for each character to 5 radices.

4) After completing a word, English typing mostly requires a space before the next word will be continued to type. In Cangjie, space is the delimiter between the radius input of two Chinese characters.

5) The wrong input code entered by typist coincidentally maps to a valid Chinese character, typists will not be aware of the errors. In English typing, nonword error will absolutely be undetected.

6) Before the space key is pressed, Cangjie allows correction on wrong radices by <Backspace> key.

2.5.2. Differences

On the other hands, there are much different aspects between the two languages:

1) Cangjie does not have the letter n-gram properties. In English words, there exists some particular patterns. For instance, the letter "u" mostly appears behind the letter "q". There is no such kind of relations in Cangjie radices. Hence, we cannot use letter n-grams analysis techniques to detect Chinese spelling errors.

2) There is no letter cases in Chinese characters, whereas we can identify proper nouns in English by identifying the words with the first letter capitalized. It may be fatal to Chinese spelling checking because proper nouns always influence the correcting capability. It is technically difficult to solve this problem by incorporate all the proper nouns in the dictionary.

3) There is no nonword errors in Cangjie-typed text as those errors will be prompted (with a "beep" warning sound) to the typists when there is no Chinese character mapped.

4) Cangjie contains repeated code which does not happen in English typing. Fortunately, the number of repeated codes in Cangjie is limited and hence predictive. In the last section, a common error dictionary approach will be proposed to solve this problem.

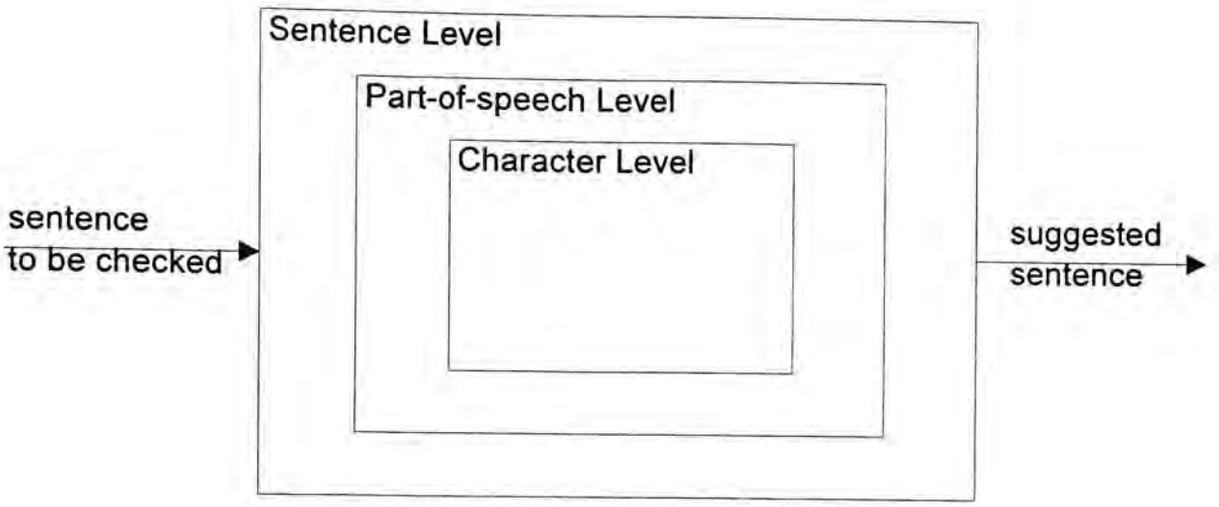
5) Once the <space> key has been pressed, the wrong radices in the wrong code cannot be corrected through the <backspace> key. It minimizes the potential deletion errors.

6) In English typing, if a number has been wrongly typed instead of the nearby letter key (for example, "r" is substituted by a wrong "4", and the word "from" will become "f4om"), it can be easily detected as nonword errors and corrected by calculating the edit-distance. In Cangjie, this type of errors will result a number in front of a character (for example, 可(一弓口) (MNR) will become "4丁" (一弓) (MN)). It may still be detected by the weird number (4) in front of a non-countable noun but cannot be easily corrected by edit-distance technique since we do not know where the ill-input letter is. It is proposed to delete the number and use similarity key techniques to correct the Chinese character which is assumed to have omission errors.

7) English word length is variable, whereas Cangjie has limited the character code length to 5 radices. It facilities a predictable size of similarity key hash tables.

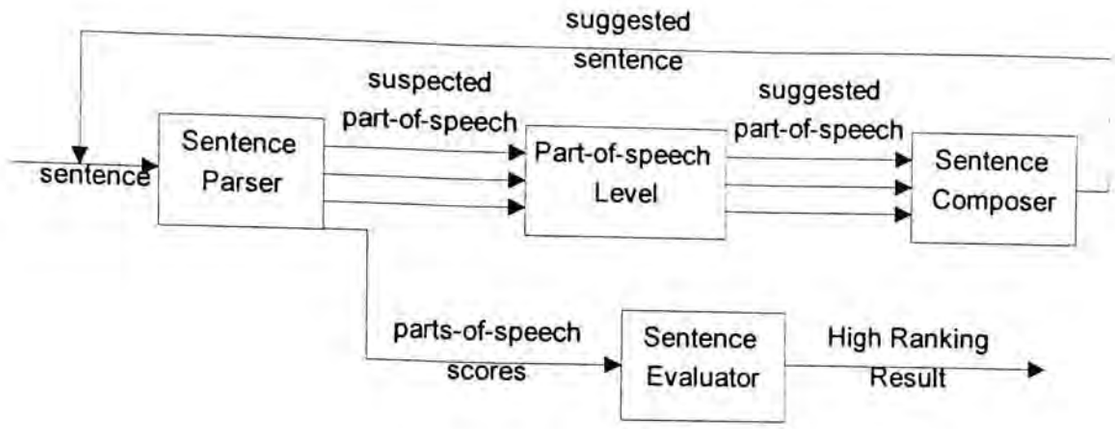
2.6. Proposed algorithm for automatic Chinese text correction

With a detailed examination on the existing technologies and comparative research on the difference between English and Chinese text, the proposed algorithm for automatic Chinese text correction will be discussed in three parts:



2.6.1. Sentence level

The algorithm will put boundaries among sentences as a detection and correction unit. The sentence to be checked will be entered into this level. The sentence parser is extracted from ACCESS. The output from the sentence parser will be also sent to the sentence evaluator. If the sentence score (average part-of-speech probability) exceeds a predefined threshold, the original sentence will be outputted. Otherwise, the suspected parts-of-speech (with low or zero frequencies) will be further sent to the part-of-speech level for correction. The suggested parts-of-speech will then be composed back to sentences and sent to the sentence evaluator for screening. The highest rank will be output. However, if all the suggested sentences score lower than the original sentence, the user will be warned for the correction failure.



Sentence parser: the parser will divide the sentence into character n-grams and determine the highest frequencies combination to produce a chain of parts-of-speech.

For Example, 我知道生命寶貴 is wrongly typed as 我佑道生命寶貴 (repeated code error).

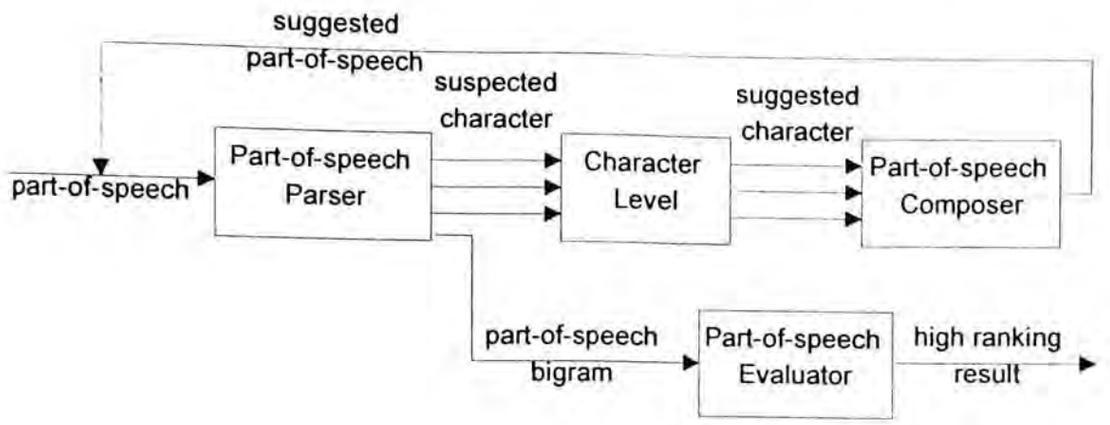
Parsing result may be:

Part-of-speech	Frequencies	
我	0.78	
佑道	0.01	(ill-formed part-of-speech)
生命	0.86	
寶貴	0.93	

Then the ill-formed part-of-speech 佑道 will be sent to sentence level.

2.6.2. Part-of-speech level

At this level, the suspected part-of-speech will be further divided into character units. Each character will be alternatively assigned as the base character of part-of-speech to detect the defected character. It is implemented with the expectation-based techniques. The suspected character will be injected into the character level for correction suggestions. Base character and the suggestions will be composed back to parts-of-speech and sent to the part-of-speech evaluator for comparisons. Higher ranking part-of-speech (either the original or the tested one) will be outputted. The evaluation process incorporates the part-of-speech bigram probabilities.



Part-of-speech parser: With the expectation-based techniques, each non-modifier character (for example, preposition is modifier, like "在" (at). The expectation list of "在" is not practical by its large size) in the sentence will be chosen as the base character. The system will obtain the expectation list of words in the previous or next position (whichever meaningful) of the base character. If the corresponding character of the original part-of-speech does not appear in the expectation list, the suspected character will be sent to the character level.

Part-of-speech evaluator: Incorporating the part-of-speech bigram probabilities, the resultant part-of-speech will be put together with the next part-of-speech to produce the corresponding probabilities. Invalid result can be detected with a very low figure under the threshold.

Let us continue on the above example, the ill-formed part-of-speech 佑道 will be further decomposed into two characters 佑 and 道. Expectation lists will then be generated from this two characters, like:

Base character	Expected character(s)
佑	護，助
道	說，知

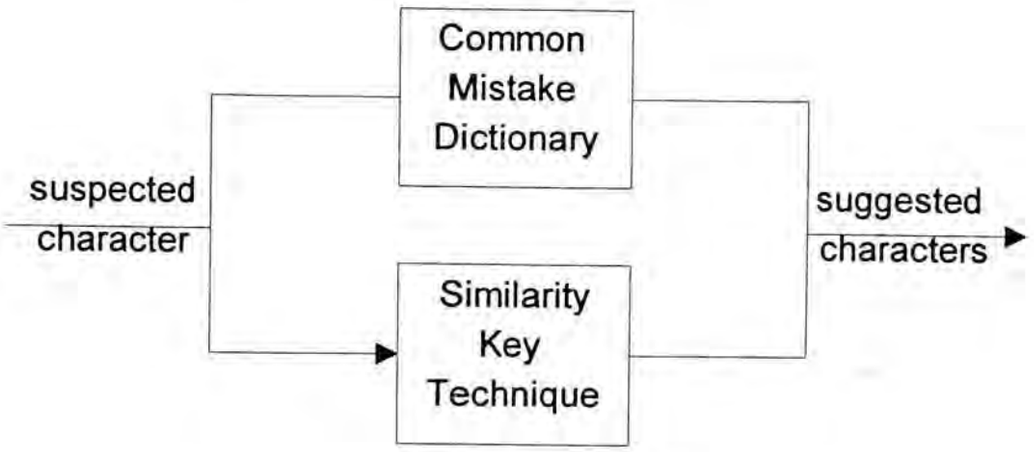
After that, 護 and 助 will sent with 佑, and 說 and 知 will be sent with 道 to the character level. In the character level, 護 and 助 will be found to be very different from 道, and also 說 will be found very different from 佑. On the other hand, 知 will be matched with 佑 by common mistake dictionary technique. The whole part-of-speech will be further weighted and found that:

Part-of-speech	Frequencies	
佑道	0.01	(ill-formed part-of-speech)
知道	0.93	(over threshold)

Therefore, the part-of-speech 知道 will be returned to the sentence level.

2.6.3. Character level

This level uses the input suspected character to generate the character alternatives for part-of-speech evaluation with the two approaches: common mistake dictionary and similarity key techniques.



Common mistake dictionary: this technique targets on the 29% of total errors created from repeated code error and character recognition / concept error. Since these kinds of errors lacking keyboarding information and may be caused by habits and conceptual problem, these errors will be generalized and spread to many documents, a (perhaps adaptive) common mistake dictionary will help to provide a solution for those characters without the need to consider the relationship with keyboard distance, context meaning or linguistic structure. This kind of dictionary cannot be aimed to be very large in size. The limited repeated code and the limited personal bias will well suit this constraint.

In the above example, 知 (人大口2) (OKR2) and 佑 (人大口1) (OKR1) will be matched in the common mistake dictionary for its repeated code property. And so the character 知 will be returned with 道 back to the part-of-speech level.

Similarity key techniques: this aims to recover the 64% of total errors including nearby key error, omission error, sequence error, extra radix error and repetitious radix error.

To implement the similarity key techniques. An sorted list of omitted radices by their frequencies should be compiled first. This list should be tailored carefully by statistical findings to achieve maximum correction power. For instance, in the research, the following list is obtained: (Note: In this research, the sample size is still relatively tiny to create an omission list. A lot more text should be analyzed to get a more accurate list.)

一口竹心廿月土大中人女日水戈田卜金木弓手尸火十山難造
M R H P T B G K L O V A E I W Y C D N Q S F J U X Z

The suspected character will be decomposed into radices, and these radices will be sorted by the above order list ascendingly and descendingly, resulted the ascending key and descending key respectively. Those keys consist of single occurrences of all radices. Examples will be illustrated later in this section.

The ascending key is used to correct the nearby key errors. Since the omission frequency of the first few radices in the list is so high, they are vital information once they have been typed. This ascending arrangement will make a closer match to the original character. The descending key is used to correct residual errors.

Here we illustrate the correction process through the two omission keys:

- nearby key error

Wrong character:	們 (人日弓) (OAN)
Ascending key:	OAN
Correct character:	但 (人日一) (OAM)
Ascending key:	OAM
Status:	OAN → OAM, close match

- omission error

Wrong character:	夕 (弓戈) (NI)
Descending key:	NI
Correct character:	名 (弓戈口) (NIR)
Descending key:	NIR
Status:	NI → NIR, close match

- sequence error

Wrong character:	釓 (金弓) (CN)
Descending key:	NC
Correct character:	小 (弓金) (NC)
Descending key:	NC
Status:	NC → NC, perfect match

- extra radix error

Wrong character:	澩 (水廿土戈) (ETGI)
Descending key:	IEGT
Correct character:	法 (水土戈) (EGI)
Descending key:	IEG
Status:	IEGT → IEG, close match

- repetitious error

Wrong character:	焱 (火火火) (FFF)
Descending key:	F
Correct character:	炎 (火火) (FF)
Descending key:	F
Status:	F → F, perfect match

Conclusion

A good interface always suggests two aspects: a friendly user interface and a convenient programming interface. ACCESS with the extendible user-friendly interface achieves these two requirements. Its high-level source codes provides good portability and stability among different machines and different versions of operating systems. X window standards with the X/Chinese toolkit reduce the burden of knowledge on Chinese computing and graphical user interface handling for normal application programmer. External function specification integrates with the external resources techniques with the contemporary programming techniques.

However, we should ask ourselves what a true user-friendly interface for Chinese system is. The word "user-friendly" in English applications cannot be put into Chinese ones directly. We should pay substantial attention on the Chinese input style since no beginners for Chinese system will admit that Chinese system is very easy to learn. Input method is still the bottleneck, and the accuracy of Chinese input will determine the usability of the system.

The above proposed algorithm will be implemented in the next phase of system development, and we are eager to see that automatic Chinese text correction will bring a true feature-enhanced user-friendly interface for Chinese systems as it can reduce users' anxiety on inputting Chinese text. In other words, it is creating user comforts.

Appendix A Cangjie Radix Table

哲理類		筆畫類		人體類		字形類	
日	四	竹	丿 厂	人	丶 𠂉	尸	彐 匚
	A	(斜)	H	亻	O	(側)	S
月	冂 一	戈	丶 厶	心	匕 七	廿	艹 艹 艹
	B	(點)	I	忄	P	(並)	T
金	丷 八	十	𠂇	手	扌 扌	山	凵 屮
	C	(交)	J	扌	Q	(仰)	U
木	丁 屮	大	乂 𠂆	口		女	𠂉 乚
	D	(叉)	K		R	(紐)	V
水	又	中	丨			田	冂
氵	E	(縱)	L			(方)	W
火	灬 小	一	工 厂			卜	丶 丷
灬	F	(橫)	M			卜	Y
土	土	弓	亅 乛				
扌	G	(鉤)	N				

Appendix B Sample Text

Article 1

(1138 Chinese characters):

【資料編輯：中山大學電算中心網路組】

以應用軟體角度看主從式架構

近來民間機構作系統轉換逐漸由集中式的電腦作業轉爲開放式的主從架構，主從式架構的特徵有分散式處理及開放式架構兩個最大特徵。分散式處理是將週期性的作業交由各部門的系統單獨處理，彙總性的資料統計分析才交由統籌單位處理，使各個系統都能在硬體設備及軟體架構上以分散式的方向處理相關資料，隨著分散式處理的作業方式衍生出硬體設備的小型化及網路連線整合，主伺服器的作業功能被多個單一作業小型的伺服器取代，所以伺服器的需求就有小型化的趨勢，而且客戶端幾乎都以個人電腦作前端處理，整個規範而言，都較以往的集中式作業的規模還小。其次是網路連線整合，在主從式架構下可能是多個伺服器共同作業，所以各伺服器間或其他週邊的連接或各區域網路的連接是藉由路由器(ROUTER)、數據機甚至於公眾網路等來達成各系統的溝通連繫，甚至可能會有開放式架構與集中式架構彼此之間的網路連線。第二個特徵則爲開放式的架構，硬體開放式架構的準則是應用軟體跨各種作業平台PLATFORM的作業方式，可移植到各作業平台，軟體作業不必變更。

從政府的白皮書公佈到民間機構的作業系統更改，可以看出主從式架構已經是目前電腦作業的主流，其間的四個大方向就是要能有分散式處理、小型化作業、各種網路連線整合，及有一個架構良好的開放式架構。

主從式架構除了上述幾點特徵外，另外一重要的因素是支援決策分析系統，對電腦作業系統而言，是以電腦儲存大量的電子資料，這些電子資料的最大目的就是專業經營決策人以各種的資料作經營決策分析。電腦由單純的資料輸入，報表處理經由圖型處理介面更進一步到成爲企業經營決策工具。

瞭解上面所述的主從式架構後，我們就可以挑選一個適合企業界使用的模式逐一導入。進行主從式架構，有幾個重點我們必須詳加控制，

最主要的條件爲：1.與企業文化背景相符合，若企業界本身電腦應用環境特殊，必須使用單一的資料庫系統架構，或是在資料保全維護上有特殊顧慮，特殊的企業文化背景是不適合以主從式架構作業。2.得到各階層的支援，不論是管理階層和使用者的支持，是作業進行的最大保障。3.人員教育訓練是極爲重要的一環，由於科技的進步異常快速，不加以訓練隨時可能遭淘汰的命運，所以教育訓練必須要成爲日常活動一般徹底執行。4.系統架構就是電腦作業的基礎，有良好的作業基礎才能構建完善的電腦作業。5.作業系統之可攜性，意味著軟體作業系統移植到其他的硬體作業平台，軟體作業系統不必更改。

潮流趨勢都是相同的，從政府大力推動跨部會的資訊整合所帶動的開放系統風潮，以及知名的大廠相繼投入主從式架構的業務，不難得知資訊應用的開放時代已經來臨，因此在此接段等待我們對開放性的主從式架構深入探討，以免落於時代潮流之外。（本文作者凱明電腦研發部經理闕申宜，資料處理左恆和）

Article 2

(1740 Chinese characters):

產品評比

選擇網路作業系統前應考慮的幾個問題

〔IDG News〕四家大的網路廠商(IBM、Microsoft、Banyan Systems和Novell)均堅持它的NOS最適合使用在企業環境下。Novell公司藉由在NetWare 4.01中增加NetWare目錄服務(NDS)而使它的功能獲得最大輻的改善。NDS直接可在DOS、Windows和OS/2需求端上使用。

Banyan擁有StreetTalk將近十年，這個技術約略類似於NDS；Vines 5.52中所提供的StreetTalkⅢ已經改善了搜尋和管理工具。

Microsoft公司使用稱爲trust relationships的延伸方式把在Advanced Server中的領域模式做了昇級處理。這可提供一個集中式的方法來管理使用者和網路資源。

只有IBM公司較爲落後：它的領域是供小型的工作群組使用，且缺乏集中式的能力。

但是，在你選擇任何一種NOS之前，你應該詢問一些你的公司以及所考慮的廠商一些問題。

目錄服務

最近在網路作業系統上所做的許多改良使得它們不僅在企業目錄上維持有關使用者名稱的資訊，同時也包含了網路上的資源的資訊，如檔案伺服器、印表機、傳眞機，以及應用程式伺服器等。理想上來說，這個目錄應該要能很容易的設定，很容易的改變，以及很容易的調整規模。

假使你還沒有建立一個全企業化的目錄，現在正是時候來研究不同的NOS，以及看看那一個對自己是最有意義的。你應該如何開始呢？首先，需要確定你的網路名稱和編號是唯一的，且你的組織有一個有系統的方法來指定一個新的物件，並且確保它是唯一的。沒有這個結構，任何產品都是沒有用的。

其次，你應該有一個全企業化的應用程式標準命名方式及存取的方法。假使在不同的樹狀結構中每一個伺服器中存放了一些不同的命名方式，則這些產品在使用上就不能夠發揮的很好。現在正是開始把程序標準化的時候，你將會發現它對於你在進一步往下規劃時有著非常大的助益。

最後你應該非常小心的設計你的目錄結構，使得它對於業務上是有意義的；我們建議你先在一小組的使用者上先做幾個月的測試，直到你已經獲得了所需要的經驗爲止。一般說來，你需要對你的初次設計做一番修改。

你的設計應該遵循地理位置的區分（例如每一個實際的地點屬於一個組織單元）或部門的區分，亦或是這兩者的合併。計劃作業中的一部份是要挑選一個易於管理的架構，例如，它能給與位於紐約的銷售辦公室內的每一個人有相同的權利和存取相同的資源。你已需要考慮那些使用者較常移動；當他們離開位置時如何存取網路上的資源；以及這些變動如何由你所選擇的NOS來處理。

你應該詢問廠商一些有關於它的NOS的問題：

■每一個伺服器均儲存有每一個使用者的帳戶記錄，或者是只有一部份的資料呢？例如，NetWare 4.01讓管理者依照他們所需要的來複製使用者帳戶，你可以決定有多少份複製儲存在何處。Vines 5.52可儲存網路上的搜尋資訊，但是只有一個單一的伺服器能夠驗證使用者：假使這個伺服器停止運轉，那麼使用者就不能登入了。

Windows NT Advanced Server是較複雜的，在一個領域內的所有伺服器均保存有所有使用者的資料，且每個伺服器每五分鐘會彼此之間對改變部份做更新。使用者的帳戶資料可以跨越領域做交換（假使管理者有設定trust relationships），但是這是一個很複雜的操作。IBM的LAN Server並沒有任何的trust特性，它只能使用在單一的領域模式中。

■除了使用者的帳戶資訊之外，還有什麼其他的資料儲存在目錄中？

■在敘述上有什麼樣的彈性呢？NDS具有最好的彈性，Advanced Server則最差；你必須決定你需要多少的彈性來建立自己的目錄資訊。

■目錄的搜尋是否容易？Vines有著最好的搜尋工具，它的StreetTalk Directory Assistance軟體能夠快速的找出企業內的任何網路資源。

變更的管理

網路的具體形式可能改變，假使有一個使用者移到新的部門，伺服器擴增了更多的磁碟機空間，或者是新的設備加入了網路中，那會發生什麼事呢？你的NOS能處理這些改變呢？或者是它會動彈不得而要求網路管理者以人工的方式進行修改？唯一能夠確定結果的方法是你實際的做設定測試，觀察需花多久時間才能完成變更。

安全性管理

雖然沒有一種網路可以完全防止有意的入侵者，但是你應該小心地看待安全性的訴求。有幾點是應該考慮的：

・所有的密碼均經過加密嗎？當NetWare或Advanced Server的Macintosh需求端使用System 7內建的Appletalk軟體登入網路時，他

們的密碼是以純文字的方法送出。假使他們使用Vines或由Microsoft和Novell所提供的附加的Mac需求端軟體，這個密碼就能以加密的方式傳送。

・是否有分開的安全性設定檔案(profile)可供網路管理者，各部門的職員，以及其他等級的管理者使用？例如，你不希望在晚上執行資料備份作業的職員存取所有的網路上的資源。所有的NOS均可提供這個功能，但是在設定時還是需要小心注意。

需求端支援

NOS是否能支援多種的需求端呢？例如，NetWare 4.01支援Macintosh需求端，但是並非NDS的一部份；Mac需求端仍需以bindery模擬的方式來執行4.X伺服器，直到Novell把功能加入它的NOS爲止。※

Article 3
(1818 Chinese characters):

小兒氣喘之發生原因探討

.作者

小兒科　劉文章教授

.本文

每當北風開始吹起，陣陣的咳嗽聲便會劃破寧靜的長夜，撩起氣喘病童無限的困擾及父母內心無涯的傷痛。漫漫長夜，經常是手足無措，不知如何是好。小兒氣喘已經是許多年輕父母最大的難題。本文謹簡單介紹小兒氣喘之流行病學，並探討其發生之主要原因以供各位家長之參考。

數十年來台灣隨著工業的發展，社會的進步及醫學的發展，各種傳染病逐漸被消滅及減少。但是氣喘的罹病率卻反有大幅增加之趨勢。自民國六十三年之一・三四%，民國七十年之五・〇七%到民國八十年之六・〇%。而同時成人氣喘也將近四・〇%。此外據一九九〇年之衛生署統計，氣喘也名列台灣地區十大死亡原因之第九名，由此可見，氣喘病乃是我國成人及小孩最常見的慢性病之一。

一般而言，小孩的氣喘其發病年齡均甚早，大約八十%左右均小於五歲，而只有十%的小孩其初次喘鳴大於十歲，其中又以男孩爲多，著者前

之調查發現其男女比約爲二：一，但隨年齡之增長，男孩及女孩之氣喘均會漸減輕，而於成人時其男女比約一：一。

愈小的氣喘小孩愈會有咳嗽，尤其是長期的夜晚半夜之咳嗽，初期時甚至未出現喘鳴，因而經常被診斷爲「支氣管炎」，或「痙攣性支氣管炎」。此種小孩如以過敏原激發或激烈運動後即可自肺功能上證明其爲氣喘。而此種經常的夜晚咳嗽或運動後咳嗽即爲幼兒支氣管喘鳴的一個特別症狀。而只對支氣管擴張劑有效，對抗生素或止咳抑咳劑卻沒效。

某些因素已經被證明與氣喘之發生有關係，例如家庭或遺傳因素、男孩、過敏體質、呼吸道感染、空氣污染、父母吸煙及支氣管之過度反應等。

（1.）家庭遺傳因素方面，調查證實父母親有氣喘的子女發生氣喘的比例有三倍於未有家族性者。同卵之雙胞胎比異卵雙胞胎之氣喘比例約爲十九%比四・八%，父母均氣喘之子女發生氣喘大約爲五十一六十%，而父或母只有一位氣喘，則子女之百分比約爲十八一十九%，而父母均無氣喘者子女氣喘之百分比則只有二・五一六・五%左右。由組織抗原之研究(HLA)也發現某些抗原有相關性。

（2.）男孩較女孩明顯：據以往之調查約爲二：一，因爲女孩之氣管阻力較小，男孩之支氣管徑較小，較易受到感染造成喘鳴。

（3.）過敏體質：過敏體質的小孩較會有較高之特異性IgE抗體，許多病兒均發現於嬰兒期時有新生兒濕疹及牛奶過敏，幼兒期時有異位性皮膚炎、過敏性鼻炎、過敏性結膜炎，而兒童期時又併發氣喘，可見過敏性的體質經常經由其特異性IgE抗體之合成，而自嬰兒期之牛奶過敏或其他食物過敏而漸轉變到兒童期的氣喘。過敏原之皮膚測驗更證明其相關性。而父母親的過敏體質更會是重要的影響因素。

（4.）呼吸道感染：嬰兒期因支氣管及小支氣管之發育未成熟，因此如遭遇下呼吸道之病毒感染，經常會造成兒童期之氣喘，著者之調查發現：於一歲以下小孩於肺炎或支氣管炎感染後五到十年有三十六・八%小孩會變成氣喘。國外的報告更高，甚至有到五十%左右。可見嬰兒期之肺部支氣管感染很可能破壞其支氣管之正常發育，而造成氣喘。

（5.）空氣污染：工業之發達造成空氣中含有各種工業廢棄物，如ＮＯ2，ＳＯ2，Ｏ3及微粒等，此種污染物將損害到幼小嬰兒之支氣管，也將造成氣喘，著者前之調查高雄學童之氣喘發生率（五・六%）遠比屏東林邊

學童之發生率（三・三%）爲低，可見一斑。

（6.）父母親吸煙：二手煙之危害的小孩氣喘有密切相關係，父母親吸煙，小孩發生氣喘之百分比接近五十%。可見如何避免嬰孩之吸二手煙，尤其是父母親能戒煙，將對子女之氣喘有幫忙。

（7.）支氣管之過度反應：某些嬰兒經常於天冷或刮風時即可有喘鳴之呼吸聲，其確實之致病原因雖未十分清楚，但此種小嬰兒長大後有較高比例發生氣喘。

如何預防小孩發生氣喘，乃是每位身爲父母者很關心之課題，本文謹提出下列方法：

一、母乳餵食並延長期間；母奶餵食可供給嬰兒很好之保護作用，可抑制IgE合成，可減少過敏病之發生，自亦可減少氣喘，尤其是父母有過敏體質者，更應該以母奶餵食小嬰兒。

二、以低過敏食品取代牛奶餵食，如未以母奶餵食也應該以低過敏食品或黃豆食品取代牛奶餵食。

三、減少環境污染，鼓勵將環境整理清潔，空氣除濕，清淨等等減少過敏原，鼓勵游泳增強體力。

四、減少早期之嚴重感染：除了母奶外，如嬰兒稍有不適，應儘快就醫，早期治療，則可減少後遺症。

五、早期檢查兒童之疾病，早期查出其過敏原，並做治療，則可減輕其病程及嚴重度。

「預防勝於治療」古有名諺，小兒氣喘病大部份均可自前述而加以預防其發生，如不幸發病也可經由與醫生之配合治療而使之不再經常的發作，減輕病兒及家長的痛苦。

Article 4

(1152 Chinese characters):

糖尿病母親所生的嬰兒

.作者1

小兒科　黃麗瑛醫師
呂志忠教授

.本文

王小弟出生時，由於體重高達四・八公斤，造成難產，只得緊急剖腹產才得以順利出生。正當全家爲他的誕生興奮不已時，小寶寶卻發生呼吸急促、抽搐的現象。經檢查才發現小寶寶血糖過低，肺部有不成熟玻璃樣膜病，因此小兒科醫師懷疑小寶寶的媽媽患有糖尿病。經追溯媽媽的病史，發現小寶寶的媽媽在懷孕中期即出現尿糖的現象，當時並不在意，也沒做定期產前檢查，因此造成此種情形發生。

糖尿病婦女懷孕時，所生的新生兒常有低血糖、呼吸窘迫及巨嬰症等問題，因此是屬於高危險妊娠。

根據美國統計，在全部懷孕婦女有〇・一%至〇・五%爲幼年型糖尿病患者，而約三%至十二%發生妊娠糖尿病，因此對於糖尿病母親所生的嬰兒，可能發生的種種問題，必須有點認識。

目前認爲糖尿病母親，所產生的高血糖經胎盤傳給胎兒，使得胎兒亦存在於高血糖環境，進而刺激胎兒胰臟小島細胞肥大，產生胎兒胰島素過高現象。胰島素的作用不止在糖份代謝，連蛋白質、脂肪等亦有影響，因而產生一連串的影響。以下就糖尿病母親所生新生兒常發生的一些現象加以敘述。

一、　低血糖：出生後由於母親供應的血糖中止，而嬰兒體內高胰島素現象仍存在，因而常在一至五小時內發生血糖過低情形。若懷孕末期血糖控制在正常範圍，發生此情形的機會則減小，一般處理方式是儘早餵食或直接給予點滴補充糖份。

二、　巨嬰症：由於胰島素會促進脂肪、蛋白質合成，因此容易產生嬰兒體重過重，雖然巨嬰症與血糖控制不良有很明顯關係，但在血糖控制良好的孕婦中亦有三〇%機會產生巨嬰症。巨嬰症也直接增加了剖腹產的機會及生產傷害，且巨嬰症的嬰兒並非對稱之增大。其肝、腎上腺、心臟特特別容易受影響而肥大，但腦則很小。另外若母親又有血管病變，造成胎盤功能不足則嬰兒體重反而容易不足。

三、 呼吸窘迫：糖尿病母親所生之新生兒較一般嬰兒發生呼吸窘迫症之機會高出五倍，主要是由於高血糖、高胰島素延遲肺部成熟。

四、 先天異常：此類新生兒較一般嬰兒發生先天異常之機會高出三至五倍。最常見的是心臟缺陷或肥大，其次如脊柱異常、中樞神經異常等均有報告。

五、 其他：此類嬰兒發生低血鈣、低血鎂的機會亦較大，還有如紅血球過多症及常合併脫水現象，因此需特別注意水份補充，以防發生栓塞，特別是腎靜脈栓塞。另外，超過二〇%的嬰兒會有稽延性黃疸，這可能是由於紅血球過多及轉換成成人血色素發生較慢的關係。另外懷孕糖尿母親，血糖控制不良亦增加自然流產或死產的機會。

綜合以上所說，糖尿病母親所生之嬰兒有許多的危險，但這些危險也常與母親血糖控制情形有很大的關係。現在醫學對於母親糖尿病的早期認知及良好控制，已明顯改善此類新生兒的死亡率及合併症的發生率。至於此此類新生兒的預後，若無其他異常情形下，發生幼年型糖尿病的機會小於二%。

Appendix C Error Statistics

1. Nearby key error

Correct	Wrong	Correct	Wrong	Correct	Wrong
者	呑	機	鐵	快	悚
生	笪	併	位	你	仔
但	們	嬰	婁	年	休
台	弋	色	笆	及	反
策	委	不	石	十	竹
就	訛	作	伯	大	十
進	塾	父	釭	甚	其
育	朗	因	夬	抑	折
恆	怔	減	滅	喘	啃
供	借	瑛	瑞	敏	倅
這	言	溯	漓	染	囝
沒	汳	低	底	清	圈
不	丁	許	訖	因	毋
它	弋	方	成	另	咋
其	萁	上	甘	黃	薰
具	賑	還	衰	綠	鶯
進	坐	立	占	放	敷
左	有	如	奴	速	桌
支	反	如	釦	應	恣
應	庇	變	娭	段	殷
年	佃	塹	在	逐	歔
溯	滿	它	毛	作	件
胎	始	除	隃	支	古
高	過	用	玳		
血	向	好	釨		

2. Omission error

Correct	Wrong	Correct	Wrong	Correct	Wrong
名	夕	用	手	機	棧
直	宜	加	力	分	刀
對	寺	個	固	止	上
網	綏	企	仩	進	仕
做	仗	方	圯	期	共
血	冊	助	肋	氣	你
伺	合	取	耳	比	庇
特	等	均	均	國	或
背	巾	照	炅	五	于
們	門	使	仗	經	煙

何	可	由	中	研	豆
清	青	高	調		
易	勿	者	丈	(2 omissions)	
呼	咪	好	子		
新	析	倍	佔	改	己
快	也	主	卜	能	弋
山	目	也	木	的	毛
需	胹	做	佔	性	必
法	圣	否	不	何	丘
不	火	率	迮	此	卜
何	合	道	逆	經	紅
愈	憊	民	艮		
有	月	底	成		

3. Repeated code error

Correct	Wrong	Correct	Wrong	Correct	Wrong
詢	詣	已	己	源	鴻
攣	姅	勢	劫	感	怠
激	淚	戒	弁	顯	顥
某	芋	知	佑	他	柒
致	玫	晚	冕	乎	釆

4. Character recognition / concept error

Correct	Wrong	Correct	Wrong	Correct	Wrong
印	卬	藉	耆	倍	倍
愈	氿	教	孝	模	摸
敏	俟	須	頸	執	報
毒	撢	問	間	同	且
鼻	算	準	汴	是	的
柬	杳	驗	騃	直	眞
順	頎	入	人	乃	及
脊	丙	任	仟	諺	訪
逐	歔	卻	郤	症	貪
業	藹	七	它	下	不
幾	以	歲	步	非	厞
通	牽	隨	隋	七	也
極	椪	父	矢	例	列
此	北	六	文	亦	弈
文	六	於	放	代	伐
減	淢	變	紋	眞	直
放	於	幸	垏		

5. Sequence error

Correct	Wrong	Correct	Wrong	Correct	Wrong
式	惑	由	甲	呼	啾
純	紀	某	苯	他	悠
小	釔	末	未	練	綑
進	住	控	擯		
應	塵	及	殳		

6. Extra radix error

Correct	Wrong	Correct	Wrong	Correct	Wrong
宜	直	化	代	工	丌
法	澀	大	有	沒	汲

7. Repetitious error

Correct	Wrong	Correct	Wrong	Correct	Wrong
三	二	出	灬	比	忿
炎	焱	例	仃	一	二

8. Mixed error

Correct	Wrong	Correct	Wrong	Correct	Wrong
架	加	小	門	織	統
與	齊	名	多	同	亙
夠	多	合	會	例	偷
例	刨	告	牡	迫	迭
們	間	發	笈	此	苑
敘	余	以	戍	外	又
結	周	植	梢	純	紀
直	疽	所	作		

9. Others

Correct	Wrong	Correct	Wrong	Correct	Wrong
(number-alphabet error)		(space between radices)		(repeated code and punctuation)	
可	4丁	相	朿山	員，	，員
斷	9斤				

References

[1] 朱邦復 <<第五代倉頡輸入法手冊>>，漢星工程有限公司，1989

[2] 周關興 <<漢字終端技術入門>>，人民郵電出版社，1990

[3] 技術資料編輯組 <<倚天中文系統技術手冊>>， 倚天資訊股份有限公司，1991

[4] 商務印書館 <<新編標準電碼本>>，商務印書館，1981

[5] 鄭茂松 <<中文軟件與軟件漢化>>，電子工業出版社，1993

[6] 戰曉蘇, 趙立軍及蘇雷 <<UNIX 系統 V 實用程序設計方法>>，科學出版社，1993

[7] **Atwell, E. and Elliott, S.** Dealing with Ill-formed English Text, *The Computatioal Analysis of English: A Corpus Based Approach*, Ch. 10, Longman, Inc., New York, 1987

[8] **Brown, C. Marlin** *Human-computer Interface Design Guidelines*, Ablex Publishing Corporation, 1988

[9] **Church, K. W. and Gale, W. A.** Enhanced Good-Turing and Cat-cal: Two New Methods for Estimating Probabilties of English Bigrams, *Computer Speech Language*, 1991

[10] **Damerau, F. J.** A Technique for Computer Detection and Correction of Spelling Errors, *Communication ACM*, Vol. 7, No. 3, Mar. 1964, p171-176

[11] **Flanagan, David** *X Toolkit Intrinsics Reference Manual*, O'Reilly & Associates, 1990

[12] **Grudin, J.** Error Patterns in Skilled and Novice Transcription Typing, *Cognitive Aspects of Skilled Typwriting*, Springer-Verlag, 1983

[13] **Kempen, G. and Vosse, T.** A Language-sensitive Text Editor for Dutch, *Proceedings of the Computers and Writing III Conference*, Apr. 1990

[14] **Kernighan, M. D., Church, K. W. and Gale, W. A.** A Spelling Correctin Program Based on A Noisy Channel Model, *Proceedings of COLING-90, The 13th International Conference on Computational Linguistics*, Vol. 2, p205-210

[15] **Kukich, Karen** Techniques for Automatically Correcting Words in Text, *ACM Computing Surveys*, Vol. 24, No. 4, Dec. 1992, p377-439

[16] **Lapin. J. E.** *Portable C and UNIX System Programming*, Prentice Hall

[17] **Lavenier, Dominique** An Integrated 2D Systolic Array for Spelling Correction, *Integration, the VLSI journal*, Vol. 15, No. 1, Jul. 1993, p97-111

[18] **Lucchesi, Claudio L. and Kowaltowski, Tomasz** Applications of Finite Automata Representing Large Vocabularies, *Software - Practice and Experience*, Vol. 23(1), Jan. 1993, p15-30

[19] **Marzal, Andres and Vidal, Enrique** Computation of Normalized Edit Distance and Applications, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15, No. 9, Sept. 1993, p926-932

[20] **McCormach, J., Asente, P. and Swick R. R.** *X Toolkit: The Intrinsics & Athena Widgets*, Silicon Press, 1990

[21] **Mor, M. and Franekel, A. S.** A Hash Code Method for Detecting and Correcting Spelling Errors, *Communication ACM*, Vol. 25, No. 12, Dec. 1982, p935-938

[22] **Morris, R. and Cherry, L. L.** Computer Detection of Typographical Errors, *IEEE Transactions on Professional Communication*, Vol. 18, No. 1, 1975, p54-63

[23] **Mullin, James K.** A Tale of Three Spelling Checkers, *Software - Practice and Experience*, Vol. 20(6), Jun. 1990, p625-630

[24] **Nye, Adrian and O'Reilly** *X Toolkit Intrinsics Programming Manual*, O'Reilly & Associates, 1992

[25] **Padovano, Michael** Motif and Open Look: two views on managing Windows. (Unix Report), *Systems Integration*, Vol. 24, No. 6, Jun. 1991, p25

[26] **Peterson, J. L.** Computer Programs for Detecting and Correcting Spelling Errors, *Communication ACM*, Vol. 23, No. 12, Dec. 1980, p676-684

[27] **Pollock, J. J. and Zamora, A.** Automatic Spelling Correction in Scientific and Scholarly Text, *Communication ACM*, Vol. 27, No. 4, Apr. 1984, p358-368

[28] **Pong, M. C. and Zhang, Y.** cxterm: A Chinese Window Emulator in the X Window System, *Software - Practice and Experience*, Vol. 22(10), Oct. 1992, p809-826

[29] **Richard, Kevin and Johnson, Eric F.** Managing X fonts, *UNIX Review*, Vol. 10, No. 2, Feb 1992, p99

[30] **Rost, Randi J.** *X and Motif Quick Reference Guide*, Digital Equipment Corporation, 1990

[31] **Rubin, Tony** *User Interface Design for Computer Systems*, Ellis Horwood Limited, 1988

[32] **Shneiderman, B.** DIrect manipulation: A step beyond programming languages, *Computer*, IEEE publication, Aug. 1983

[33] **Shneiderman, Ben** *Dessigning The User Interface: Strategies For Effective Human-Computer Interaction*, Addison-Wesley, 1992

[34] **Suen, Sunny T.** *X/Chinese Toolkit*, Dept. of Computing, Hong Kong Polytechnic, 1993

[35] **Turba, T. N.** Checking for Spelling and Typographical Errors in Computer-based Text, *SIGPLAN-SIGOA Newsletter*, Jun. 1981, p51-60

[36] **Wagner, R. A.** Order-n Correction for Regular Languages, *Communication ACM*, Vol. 17, No. 5, May 1974, p265-268

[37] **Yannakoudakis, E. J. and Fawthrop, D.** An Intelligent Spelling Corrector, *British Journal of Educ. Tech.*, Vol. 22, No. 2, Feb. 1991, p146-148

[38] **Young, Douglas A.** *The X Window System: Programming and Applications with Xt*, Prentice-Hall, 1990

[39] **Zamora, E., Pollock, J. J. and Zamora, A.** The Use of Trigram Analysis for Spelling Error Detection, *Information Processing in Management*, Vol. 17, No. 6, 1981, p305-316