漢字字碼與資料庫國際研討會, 京都· 東京, 1996 年 10 月

從缺字問題，談漢字交換碼的重新設計——第二部份

A Descriptive Method for Re-engineering Hanzi Information Interchange Codes

謝清俊

中央研究院 資訊科學研究所 文獻處理實驗室

1996 年 10 月 4 日

A Descriptive Method for Re-engineering Hanzi Information Interchange Codes

1.Criticisms of Existing Interchange Codes

Every body agrees that the existing Hanzi Interchange Code is not good enough for daily applications. It seems never satisfy users requirement. As described in Part 1, obviously, the missing character problem is a fatal evidence of the insufficiency of the existing Hanzi Interchange Codes. The consequences of missing character is severe. It seems that the only solution to the problem is to re-engineering the existing coding system for Hanzi.

It is interest to notice that the meaning of " missing character" is not clearly defined. It is very fuzzy what is missing, a character, a glyph, or a typeface ? This question leads to another drawback of interchange code. That is the existing code do not distinguish character and glyph. Although there are definitions for character and glyph stated in document, but they seem never been faithfully implemented. For example, there are two code positions 饑（C4C8） and 飢（B047） in Big-5 Code. Are they representing different characters? By definition, they should represent different characters because they have different code words. But, in fact, they are merely two glyphs of a character. This situation is common to all existing codes without an exception.

One may argue that there are similar situations in ISO 646, such as the capital and the lower case of alphabets. For alphabets, it is true that there are two set of glyphs in ISO 646, but don't miss the fact that they also share the same collating sequence in ISO 646. In Hanzi codes, there is no such structure to show the corresponding collating sequence between variants, except CCCII and EACC. Therefore, a structure which maps a character to its associated glyphs, or variants, should be added to the existing coding structure.

Another drawback of the existing coding structure is the implied assumption that the set of Hanzi is a closed finite set just like that of alphabets. Theoretically speaking , this may be true as long as the Chinese language do not produce new characters ever since. But in real life, this assumption is too strong to be true and not practical at all.

Various reasons are given in Part 1. Besides, so far, no one knows how many characters are there for Hanzi. Therefore, from engineering point of view, an open coding structure is required to accommodate the characteristics of Hanzi set. Otherwise, unavoidably, the missing character problem will never be eliminated, no matter how diligent we try to collect "all" of the possible glyphs that may be founded in numerous documents.

The central theme of solving the missing character problem is to represent more knowledge of Hanzi, especially the knowledge about glyph, into computer so that computer can use related knowledge to do what we want it to do. By doing so, code must be changed , because Hanzi code is a major knowledge structure for Hanzi in computer at presentation situation. Besides, in order to improve the performance of the existing Hanzi processing system for possible applications in the future, the following principles should be considered.

1 · **Do not sacrifice Hanzi information sharing for solving coding or missing character problem.**

2 · **The solution should be fair to all Hanzi used in different countries and regions.**

3 · **Establish the capability to represent, to input, to search, to share and to manage missing characters.**

4 · **Give formal working definitions to character, glyph, font, typeface, etc. so that they are acceptable to linguistics and Wen-zi-xue. Formalize their relations.**

5 · **Establish a database for the attributes of Hanzi.**

6 · **Use ISO 8879 SGML to describe missing glyphs and Hanzi text files for text sharing.**

7 · **Expandability and flexibility of the system must be considered for further adopting traditional Wen-zi-xue knowledge into the system.**

8 · **Keep the working environment stay in two-byte code space so that the existing application software can still apply.**

2. The Representation of Hanzi Knowledge

(1)Previous Studies

(a) A Fundamental Character Set for Computer Use

Around the year of 1970, the study of using computer to process Hanzi information have been launched in Taiwan. During these days, the statistical knowledge of characters and words is not enough to support research need. Therefore, Mr. Su Lin of the Computer and Control Engineering Department of National Chiao-Tung University conducted a research on finding " A Fundamental Character Set for Computer Use" with the support of the Wang Laboratories. The research started in October of 1971, spent more than 2000 man-days, and a draft report was published in

March, 1972. Some highlights of this research are listed as follows.

1 · **A thorough survey of all the statistical studies of Hanzi from 1856 to 1971had been carried out. And a survey of character sets used by popular dictionaries and press media during that period had also been done.**

2 · **11 character sets surveyed in item 1 were selected and weighted. Their union set was named as the Fundamental Character Set for Computer Use (中文電腦基本用字). A list of these 11 sets are shown in 〔Table 1〕. These 11 sets actually covers more than 30 previous statistical works on character set..**

3 · **Variants are also collected. References about variants including 林語堂〈整理漢字草案〉 and 國立編譯館《常用字統一字形暫用表》（1968）. Some guidelines for selecting authority glyph are listed in 〔Table 2〕. Other glyphs, or variants, are also collected as "参照字形". So, there are simplified characters in the collected character set. In general, this research does not care whether the glyphs are right or wrong, complicated or simplified, normal or popular, ancient or up to date. The collection is merely aiming at all possible glyphs that computer might possibly confront. This altitude is not the same as that of all previous work.**

4 · **This research collected 8532 characters, and with additional 597variants. The total frequency count of usage is 2,022,604, and their distribution are listed as follows ：**

category	Number of characters	percentage of use
most frequently used	1857 字,	97.34%
frequently used	2068 字,	2.27%
occasional used	2182 字,	0.27%
rare used	2425 字,	0.12%

The entropy of the system is 9.60. The accumulated frequency of the most frequently used 500 characters are listed in 〔Table 3〕. Up to now, this survey is still the most comprehensive one and we use it as a foundation of our study.

Table 1 : The 11 character sets collected in 《中文電腦基本用字集》

1. 1. 莊澤宣,《基本字彙》, 廣州中山大學教育學研究所, 1930
2. 胡顏立,《小學初級分級暫用字彙》教育部, 1935
3. 教育部,《注音漢字》商務印書館, 1935 初版, 1961 台一版
4. 蔡樂生,《常用字選》英文中國郵報社, 1946
5. 台灣省國語推行委員會,《國音標彙編》, 開明書局, 1947 初版, 1971 台二版
6. 王清波,《國民小學現行國語課本國字初現課次、重現次數之分析研究》, 高雄市政府, 1963
7. 國立編譯館,《國民小學常用字彙研究》中華書局, 1967
8. 台灣電信局,《電碼新編》, 1967 增訂版
9. 星華打字儀器行,《中文打字機新版文字排列表》, 台北, 1969
10. 世界中文報業協會,《新聞常用字彙》, 1970
11. 中南鑄字廠,《常用字表》, 台北, 1971

Table 2: 《中文電腦基本用字集》異體字的整理原則

1. 就已有字彙選取，不另創新字。
2. 一字數形, 取其簡便者。而不計其本體抑俗體, 古字抑今字。古字簡便者從古，如取「　」不取「禮」, 今之簡便者從今，如取「　」不取「繡」。
3. 一字數形，取其結構適合電腦設計者。如取「略」不取「　」，取「裡」不取「裏」。
4. 一字數形，取其通用者，如取「拿」不取「拏」。
5. 在世俗上已通行一體，而原字還有其他意義的，則兩者並存。如「尿」、「溺」。

Table 3 : Accumulated frequency of 《中文電腦基本用字集》

number of characters	accumulated frequency	number of characters	accumulated frequency	number of characters	accumulated frequency
5	9.24%	50	32.39%	372	+70%
10	14.20%	60	35.22%	472	+75%
15	17.79%	80	39.92%	500	76.27%
20	20.54%	100	43.74%	1000	89.38%
30	25.13%	141	+50%		
40	29.02%	232	+60%		

Note ： The sign "＋" indicates「just over」. For example ＋ 50% means the accumulated frequency of the first 141characters is just over 50%.

(b)The Chiao-Tung Root System

The earliest study of glyph structure in Taiwan was carried out in National Chiao-Tung University. Thus, the system developed was named as the Chiao-Tung Root System. In 1972, a master dissertation of Mr. 倪耿,《中國文字之結構模式及其分析》 analyzed 16 different compositional operators of glyph and found that only three of them, namely horizontal composition, vertical composition and contain composition, are frequently used and can be assembled as an effective system for representing the structure of glyphs. A formal representation of the system in Bakcus Normal Form is shown in 〔 Table 4 〕.

Table 4 : A formal representation of Hanzi Glyph in Bakcus Normal Form

〈 character set 〉 ：：＝ 〈 glyph 〉 ／ 〈 symbols 〉
〈 symbols 〉 ：：＝ including punctuation symbols, pronunciation symbols, and others
〈 glyph 〉 ：：＝ 〈 root 〉 ／ 〈 component 〉 ／ 〈 glyph 〉 〈 operator 〉 〈 glyph 〉
〈 component 〉 ：：＝ 〈 root 〉 ／ 〈 component 〉 〈 operator 〉 〈 component 〉
〈 operator 〉 ：：＝ horizontal 、 vertical 、 and contain
〈 root 〉 ：：＝ there are 496 roots as shown in 〔 Table 5 〕

Mathematically speaking, the system in 〔 Table 4 〕 is a production system. That means the possible produced outcomes, such as character, glyph and component usually are far more than we may accepted. Whether the outcome is a legal item or not is up to our choice. Thus, this system has the property of expandability that just fit our need.

In this productive system, the structure of a glyph is expressed as an expression of roots. Root is a basic component which will not be decomposed further. An example of glyph structure is shown in 〔 Figure 1 〕. The term component is usually used to reference the intermediate parts between a glyph and its roots. In 〔 Figure 1 〕, 灣 and 彎 are glyphs,䜌 is a component, 弓、言、系 are glyphs and are roots also, 氵 is a root 。

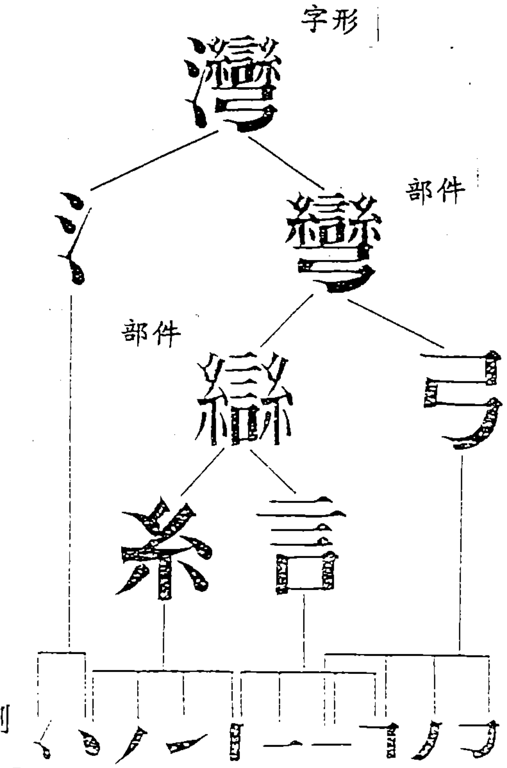
The work of Mr.倪耿 were done parallel with the work of 《中文電腦基本用字集》 by Mr. Su Lin, and the set of Chiao-Tung Roots was found by them. This root set has a unique figure that it is obtained by three times iteration of an optimization

Table 5　中文字根表 (依頻率高低排列)

口	亻	日	白	儿	辶	門	木	一	言	氵	女	月	宀	人	文	扌	也	囗	丬	卩	足	才	豕	彳	入
走	小	ナ	寸	攵	糸	我	又	夕	貝	亅	目	十	田	禾	夂	刂	心	大	力	广	匕	八	可	厶	立
上	方	王	巾	在	⺈	西	斤	竹	罒	冖	來	石	頁	古	礻	里	圭	二	隹	殳	土	至	工	灬	忄
止	車	生	𦍌	虫	去	衤	亠	尸	𦥑	金	艮	冂	尔	火	户	尤	中	勹	巴	矢	千	乍	天	欠	厂
⺗	弓	用	犬	于	馬	已	几	戈	丷	㇆	亼	者	山	耳	臼	幺	自	戊	三	豆	士	為	釆	𧾷	下
毋	彡	少	牛	五	乂	事	正	罒	雨	看	亶	水	⺌	堇	其	弟	爿	永	石	䒑	𧘇	发	开	云	且
手	直	長	夲	丁	叀	皮	勿	之	聿	面	先	门	冫	北	六	犭	刀	了	皿	民	亥	非	𠂉	孑	亡
尹	东	夕	吕	卜	癶	㡭	亻	氏	七	冉	兩	四	户	九	羊	而	東	夫	由	夬	亚	無	己	告	良
臣	厷	气	マ	身	疒	奎	丸	高	舟	牙	未	黃	及	才	头	匚	夊	电	世	叉	必	更	酉	甬	𡗗
㐅	太	求	𠂢	㔾	乃	予	魚	氺	廾	廿	巴	曲	角	声	婁	几	爪	内	刂	艮	丈	凵	入	肀	小
黑	斗	甫	羽	申	旅	丰	乚	曰	丨	韋	犮	央	辰	共	关	半	争	刃	制	坐	子	鬼	冎	片	帶
非	丘	並	毛	東	夗	州	兆	飛	弋	办	无	甘	为	虍	矛	巠	㒳	尺	皁	垂	革	弗	內	丯	缶
ト	丄	凡	束	末	午	乎	毌	川	久	戋	乡	甲	曷	柬	风	肉	镸	鬥	E	骨	具	弌	㇄	刍	瓜
耒	𠬝	乘	巨	户	甴	小	丩	戉	乚	承	爪	朮	屯	瓦	壽	乍	爭	史	莫	尞	く	平	乙	段	丹
匀	麗	夊	朿	兼	互	丞	龍	丂	疋	册	烏	业	曹	幸	肅	氺	井	阝	典	豸	卯	亙	土	重	尸
屮	亇	喪	表	牽	斥	衰	甚	与	乇	丐	爽	冘	弔	申	去	力	互	函	鹵	巫	亢	耑	两	黽	叙
𠂤	包	隺	乐	㇇	才	與	电	民	产	内	幾	※	丿(上撇)	丿(左上撇)	丶(上點)	丶(右下點)	芈	少	毌	臣	丢	爾	叉	厶	生
尸	尢	冂	豖	乖	隶																				
☆	的	是	有	他	這	國	們	說	個	就	要	全	到	以	你	時	那	裡	和	道	得	家	麼	後	樣
#	盟	丏	孓	乱	凮	凸	業	帍	华	爲	卍	叉	黽	戔	コ	乜	囙	奭	冓	冄	巛	网	ㄣ		

說明：1. 本表依字根出現頻率之高低由左而右，由上而下順序排列。
　　　2. ☆爲酌留 "常" 用字"井" 爲罕用字根。
　　　3. 本表計收字根 448 個，酌留常用字25個，罕用字根23個，總計 496 個。

Figure 1



procedure. The optimization procedure is derived from a mathematical calculation of optimizing a polynomial expression of the total number of the roots and the averaged number of roots per glyph. In general, the less the number of roots, the longer the decomposition of glyph. The optimization produce an criteria as follows : a glyph should not be decomposed if its frequency of usage is over 0.3758%, should be decomposed into no more than 2 roots if its frequency is from 0.1879% to 0.3758%, no more than three roots from 0.1236% to 0.1879%, and no more than 4 roots from 0.0939% to 0.1236%【 1 】. This criteria determined the bottom line of decomposition.

According to the 9132 glyphs in《中文電腦基本用字表》, 496 roots are obtained as listed in 〔Table 5〕. In this root set, 305 roots are characters and the frequency of usage of them exceed 50% of the total usage. The accumulated frequency of the most frequently used 25 roots is 30%, for 50 roots increased to 49%, 100 roots to 66,7%, 200 roots to 84.9%, and 300 roots up to 95%. 【 2 】

As a result of optimization, the weighted average of number of roots per glyph is only 1.9 。And the power of this system can be illustrated by showing that while checking against the 49905 characters of《中文大辭典》, 48713 characters can be expressed by the Chiao-Tung Root system. The remaining 1129 characters are 籀文, 篆字, or some ancient 古文、反文、圖騰 etc. If needed, these 1129 glyphs can be included into the Chiao-Tung Root System at any time without difficulty. A pictorial illustration of this result is shown in 〔Figure 2〕.

The Chiao-Tung Root System is developed according to the 楷書 font. Therefore, it does not include the font of 篆、隸、行、草, etc. It does not taking account of calligraphic variants , print fonts and the modern artistic fonts, neither. All it can provide is the common structure of glyph which is the basis of every font design.

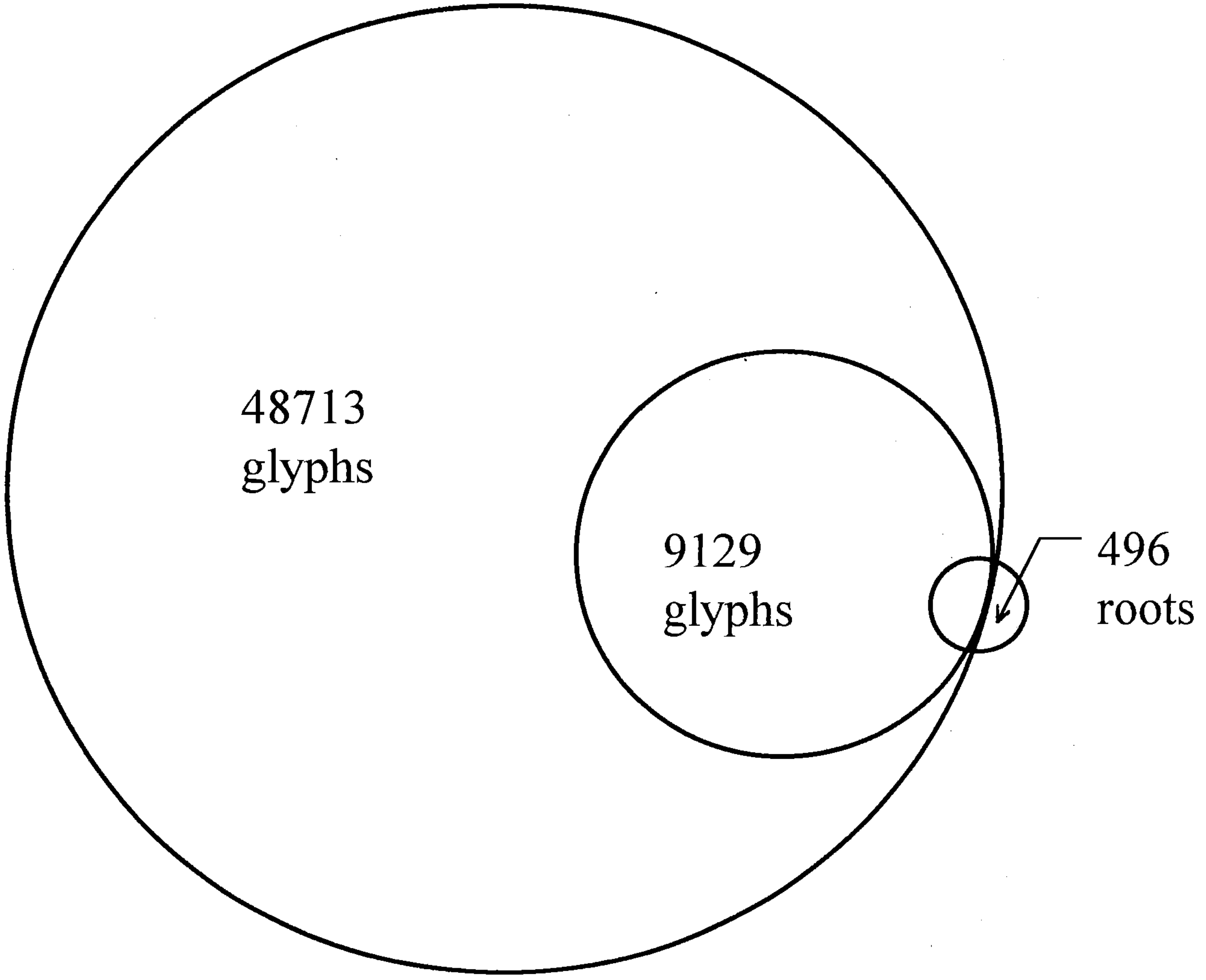
(2). Definitions

In this section, some definitions of basic terms will be given as a basis for further development. These definitions follows the discussions in Part 1.

【1】 此邊際效用之計算，請參考：謝清俊、黃永文、林樹，《中文字根之分析》交大學刊，第六卷·第一期，1973 年 2 月

【2】 關於 9129 個字形之分解及字根之使用頻度之資料，請參閱劉達人、杜敏文、謝清俊、張仲陶、蔡中川、林樹《漢字綜合索引字典》 Asian Associates, Bedford, New York 1979

Figure2. 496 roots obtained from 9129 glyphs can produce 48713 glyphs and more.
(A study of 1972 by 〔 1 〕)



Character is an abstract concept. They are differentiated only by the meanings they carried. For example, a normal form and its corresponding simplified form are considered to be the same character. Characters are identified only by the code assigned to each of them.

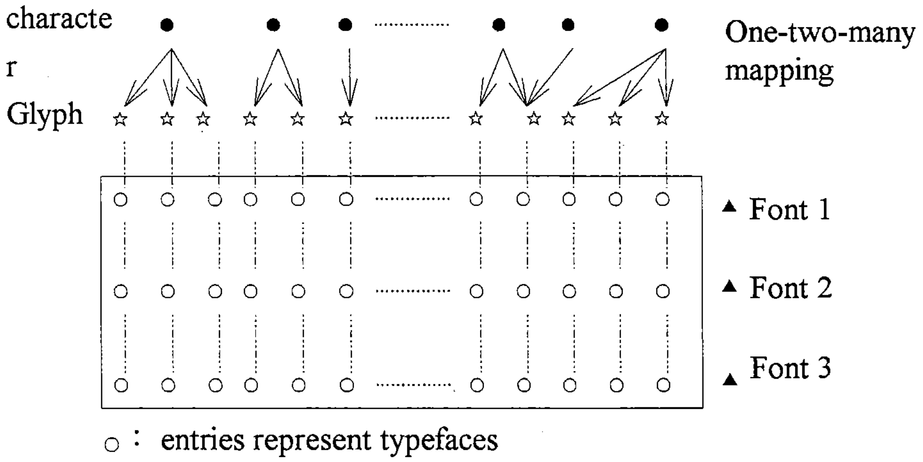
A character may have many glyphs. Glyph is also an abstract concept. They are differentiated by their own structures or skeletons. As in the previous example, the normal form and the simplified form of a character are two glyphs of that character.

It is possible that some characters share one glyph. Since we do not dealing the semantic meaning of characters at this moment, we postpone this problem as a future research topic.

Glyph does not care how nice a typeface may shown, but font cares. A font is a collection of general design rules for a certain style of typefaces. So, font is also abstract in sense. They are differentiated by their design rules. Although font specifies design rules, it has some degree of freedom that allow designers to express their own posture. A font designed may have some parameters to specify the size, the broadness of strokes, the ratio of broadness of vertical and horizontal strokes, etc.. After these parameters have been chosen, then one can see a physical typeface on media.

The described relationship among character, glyph, font, and typeface are shown in 〔Figure 3〕. Please notice that existing Interchange Codes and software systems do not and can not differentiate character and glyph. This phenomena is the source of all troubles of Hanzi processing that we have today.

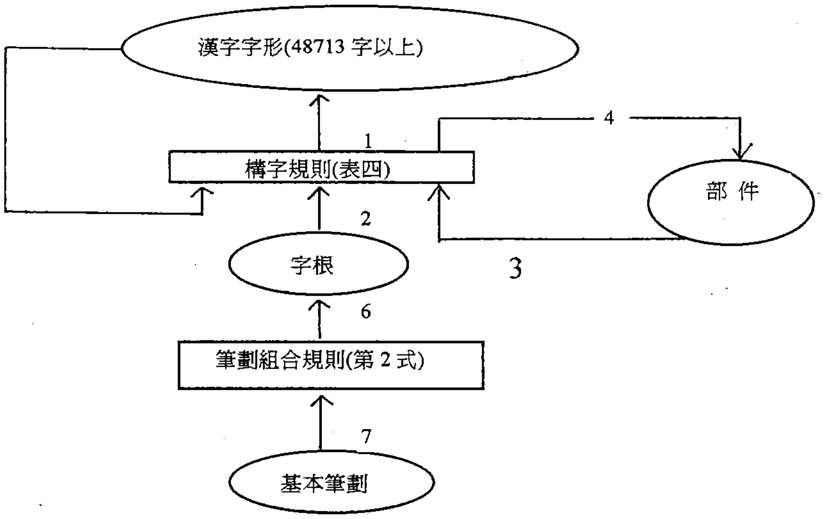
Figure 3 :
The relation among Character, glyph, Font and typeface



(3)A Glyph Model

Our glyph model is shown in 〔Figure 4〕. In this model, the part connected by line 12345is the formal system shown in 〔Table 4〕。Lines 6 to 7 show the composition of strokes to produce root. The decomposition of glyph into components can be expressed as in the following two equations.

Figure 4



Let G be glyph, R be root, K be component, and T be basic stroke. Let p and s represent position and size , respectively, then

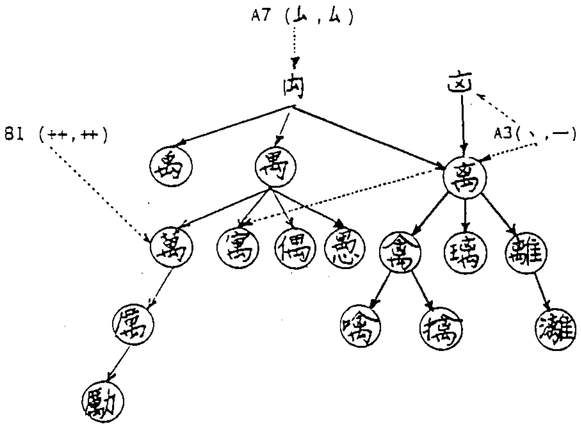
$$G=\Sigma K\ (p, s)\ \dots\dots\dots\dots\ (1)$$
$$R=\Sigma T\ (p, s)\ \dots\dots\dots\dots\ (2)$$

Equation (1) is iterative and should be governed by the rules in 〔Table 4〕。

(a)The Representation of Glyph Structure

Let us explain how the structure of a glyph is represented in computer by examples. In the tree shown below, the glyphs in right most branch can be expressed as

1.灕＝氵⿰離
2.離＝离⿰隹
3.璃＝王⿰离
4.擒＝扌⿰禽
5.噙＝口⿰禽
6.离＝㐫⿱内
7.禽＝人⿱离
8.隹＝亻⿰⿰圭
9.圭＝亠⿱主
10.㐫＝亠⿱凶
11.凶＝凵⿶㐅



These equations are generally referenced as “glyph structure expressions”, or “glyph expressions” in short, in which ⿱, ⿰, and ⿴ represent the operations of “vertical composition”, “horizontal composition”, and “contain composition”, respectively. Although there is only one composition operator in each of the above equations, computer can iterate these expressions to obtain an expression solely with roots by eliminate components successively like follows.

灕＝氵⿰（离⿰隹）
　＝氵⿰（（㐫⿱内）⿰（亻⿰主））
　＝氵⿰（（（亠⿱凶）⿰内）⿰（亻⿰（亠⿰主）））
　＝氵⿰（（（亠⿱（凵⿴ㄨ））⿰内）⿰（亻⿰（亠⿰主）））……….(3)

In Equation 3, the glyph “灕” is composted up by 8 roots. The glyph expression , such as Equation 3, solely by roots is called a “root expression”. Similarly, a glyph expression by components such as equation 1 to 11 described previously may be called a “component expression”. When all the operators are eliminated, Equation (3) becomes.

灕＝氵亠ㄨ凵内亻亠主 ……………………………………………………(4)

Equation (4) is called “the root sequence” of 灕. Following the same naming thought, equations 1 to 11 after eliminating operators may be called “component sequences”.

In general, any representation called “expression” refers to a complete glyph structure information, while “sequence” does not. In other words, each glyph has an unique glyph expression and hence this expression can be served as the identifier of that glyph. Although “sequence” does not have complete structure information of glyph, it still has very high discriminating abilities among glyphs. For instance, in 林樹字集 of 9129 glyphs, there are only 8 pairs of glyphs have exactly the same component sequence, such as （唄、員）. All others can be uniquely identified by their component sequences.

In practice, all of the glyph expressions of a character set can be stored in computer. By doing this, all the knowledge of glyph structure information of a character set has been formally represented and stored for further use. For《中文電腦基本用字》, there are 9756 glyph expressions in which 8529 for authority character, 593 for variants, and 629 for components. While simplified characters are included, then, 2284 expressions are added for simplified variants and 35 are added for simplified new components. Thus, the number of component increased to 644, and the total number of glyph expressions increased to 11477. The 8529 expressions for authority characters remain unchanged.

(4)Markup Tags for Missing Character and Variant

(a) Kanji Placeholder

Applying SGML tag to markup missing character was first used by Wittern and App.【 3 】 The technique they developed is called Kanji Placeholder (漢字位標, 簡稱位標）。Kanji Placeholder starts with " & " and closes with " ; " , and there are two fields in-between. The first field is an code identifier and the second field is a code word at which the missing glyph was found. Kanji Placeholder provides a linkage crossing Interchange Codes to share glyphs.

As an example,"&U4AB5;" represent a missing glyph found at location 4AB5 of Unicode which is identified by the field "U". Kanji Placeholder is helpful to share glyphs collected by various Interchange Codes provided that there are a collection of Interchange Codes accessible be user. Wittern and App do have a Kanji Base on Internet which collects many Interchange Codes, such as CNS of more than 45 thousands glyphs and Unicode of approximately 22 thousands glyphs, etc. Besides, they also build a bank for missing characters which can not be found in any of the Interchange Codes they collected.

(b)Hanzi Glyphholder

Kanji Placeholder assigns SGML tag as the carrier of a pointer indicating the location of missing glyph. We extended their idea by applying SGML tag to show the structure of the missing glyph and use it as the identifier of the missing glyph also. The technique is called "Kanji Glyphholder".

In order to avoid using any graphic symbol of code word as control symbol, special symbols 形 and ⊡ are created to represent open delimiter and close delimiter of the glyphholder tag, respectively. In the carrier formed by glyph holder tags is the glyph expression of missing glyph. For the convenience of use, component sequence is allowed to replace glyph expression as long as there is no ambiguous happened. For example, in Buddhist Canon 阿門佛 can be expressed as 阿形門人人人⊡佛, or 阿形門◯⊡佛.

The glyph holder can also be used to represent a variant by applying the glyph code of the "three segment coding scheme" of a character described in the later section. For example, "芍形藥・ 3 ⊡" represent "芍葯", if 葯 is the third variant of 藥.

【 3 】 Christian Wittern and Urs App.〈 IRIZ Kanji Base : A New Strategy for Dealing with Missing Chinese Characters 〉世界電子佛典會議(EBTI)台北, 1996 年 4 月

Glyphholder and placeholder are compatible with each other, because they share the same tagging structure. The two fields of the placeholder can also be accepted in glyphholder if a parser is designed to recognized them. The glyphholder has some interesting properties that the placeholder can not provide. For instance, the glyphholder is more readable than placeholder, the user is not required to looking for the missing glyph elsewhere in order to obtain an identifier or a code word for the missing character, the user's front end is not required to equipped with a data base of collecting various glyphs and variants, and finally, the glyphholder can express the mapping relation between character and glyph.

(5)Attributes

The attributes we collected so far for a character are listed in the following table.

<u>表六　文字屬性欄位表</u>　　（註:打"*"者，可以重複）

甲、缺字屬性表

1. 缺字統一編號
2. 交換碼
3. 內碼(造字檔內)
* 4. 部首
* 5. 筆劃數
6. 首筆
7. 次筆
8. 末筆
* 9. 注音
*10. 異體字交換碼
*11. 登錄日期及修改記錄
*12. 提供缺字之各單位欄位（含編號及內碼）

乙、字形結構屬性表

1.所屬字集編號
2.交換碼
3.字形碼
* 4.部首
* 5.筆劃
6.首筆
7.分解方式
8.部件一
9.部件二
10.部件三
11.字頻次
12.字根頻次(當用爲字根時)
13.字根次(當用爲字根時)

3. Design Issues

(1)Three Segmented Code Word

In order to provide a mapping among a character, its associate variants and possible selection of font types, a three segmented code word for Hanzi is designed as follows:

< a code word> ::= < Character code> (. Glyph extension) (- Font id.)

where parenthesis indicate optional terms. For example, a code word of a Hanzi can be either one shown as follows,

a character code of a Hanzi is simply : < a character code>
a glyph code of a Hanzi is : < a character code>. < glyph extension>
a font code of a Hanzi is : < a character code>- < font id.>
a style code of a Hanzi is : <a character code>.<glyph extension>-<font id.>
or, < a character code>-<font id.>.<glyph extension>

The data type of <glyph extension> is simply a positive integer N which indicates the Nth variant of a character. The <font id.> is an identifier of a certain type of font. And, the < a character code > can be any existing Hanzi Code or a new code as we proposed later.

The three segmented coding scheme of Hanzi can be used in conjunction with the glyph expression described earlier. In this case the character code can be replaced by glyph expression or component sequence of a glyph. Besides, the glyph extension can also be coded to identify various character sets, such as assigning glyph extension equals to 2 for simplified characters. Thus we can code more knowledge about characters into the system.

(2)Fidelity Levels

Another advantage of the three segmented coding scheme is its flexibility to provide different levels of fidelity for various application requirements. The fidelity levels are:

1 · **The fidelity level is the lowest when only character code is used. In this case, the application does not care which glyph or which font is been used. For example, 台灣 and 臺灣 are the same. The only matter it concerned is the meaning that the text carried. Many application can be satisfied at this level.**

2 · **The next higher fidelity level is the situation where glyph code is been used. In this level, correct structure of glyphs is required, but not font.**

3 · **While font code is been used, correct font is required, but not the glyph.**

4 · **When style code is been used, correct glyph and correct font are required. The fidelity level is higher the those of item 2 and 3.**

5 · **When some parameters of font have been specified as an extension of the font code, such as using font tags of HTML 3.2, Netscape 3.0, or Microsoft Explorer 3.0, then, the system provide the highest fidelity level.**

(3)Glyph Structure as Code Word

In previous sections, it is shown that each glyph has an unique glyph structure expression and hence it can be used as the identifier of the glyph. In this section, a new coding scheme for Hanzi interchange based upon glyph structure will be presented.

There are several advantages to use glyph structure as a foundation of designing Hanzi Interchange Code. Firstly, the component set as well as the root set of Hanzi is a closed system. It will not expand indefinitely as the character set does. Although there are rare chances that they might expand, they are far more manageable then that of character set.

For reference, consider the root set developed at Chiao-Tung University in 1972, the 496 roots derived from a set of 9132 glyphs actually can produce 48713 glyphs in a bigger collection without introducing any new root. The second advantage is that the glyph structure model is a productive system, in mathematical terms. This means it has the expandability and flexibility of not changing the existing system over newly created characters. Thirdly, glyph structure is a kink of knowledge representation. It no only facilities character coding and human reading, but also provide more knowledge for further information processing applications.

(4) The Glyph Bata Base

A glyph data base is implemented in IBM compatible PC with Window 3.1 operating system and with Chinese Extension (Taiwan Version). Programs are developed by Visual Basic language and databases by ProFox DBMS.

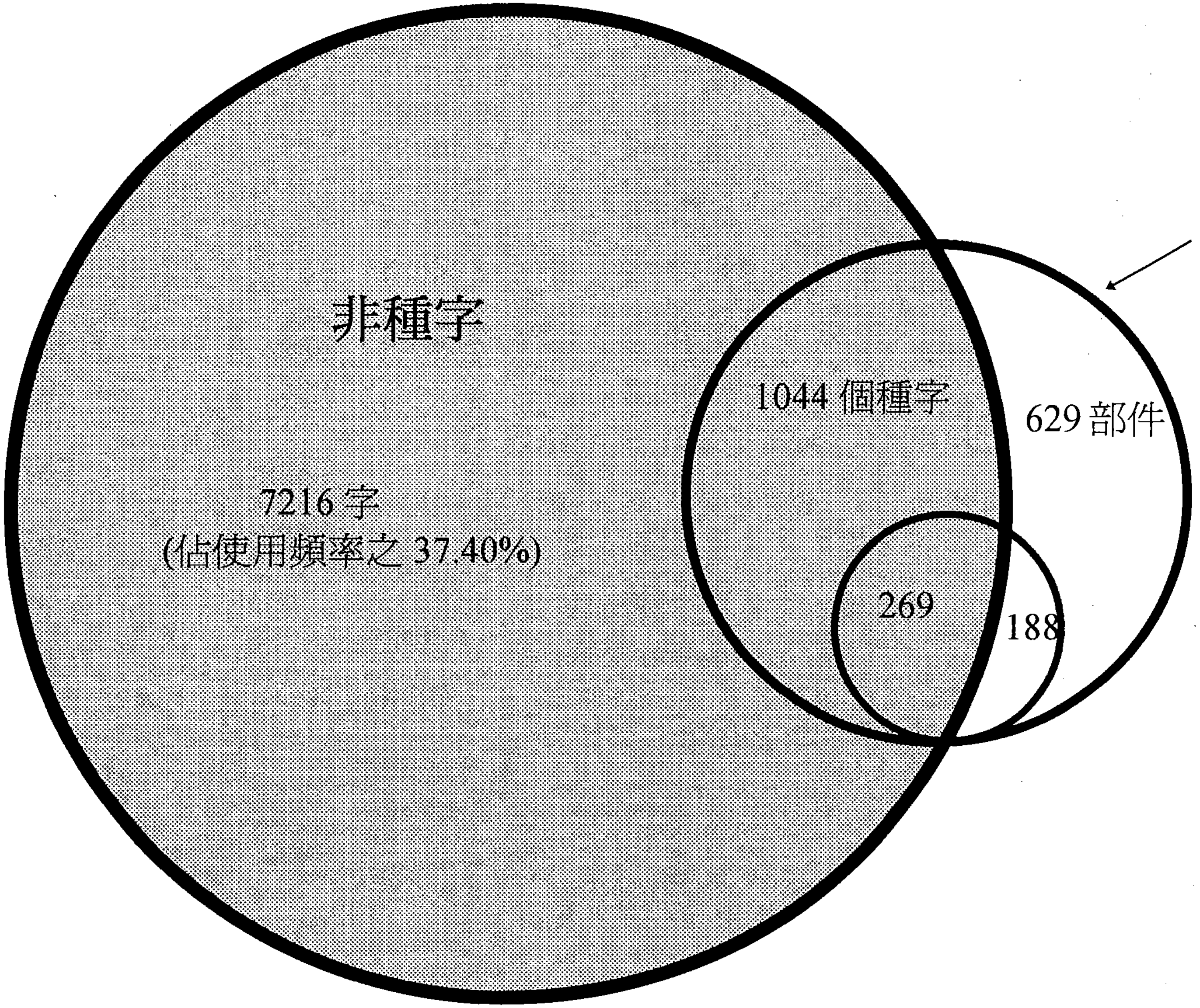
So far there three character sets already built in our system. (1) A set of fundamental Characters for Computer use《中文電腦基本用字》. This set collected 8529 characters with 593 variants. In its glyph model, there are 629 components, 457 roots。 A pictorial illustration of this set is shown in 〔Figure 3 〕. (2). An extension of the formal set by including simplified characters used in PRC. The character count does not increase , but the number of variant increased to 2284 and the component increased to 664, the number of roots to 492. (3).The third set is a supplement character set for Buddhist Text《電子佛典補充字集》. Now , more than 2000 missing characters are collected in this set. The member is glowing day by day.

(5)The Kernel Set

As component expression is used as an identifier or a code word for a glyph, the average number of component per glyph has to be minimized. This can be done by limiting the operators in a component expression to be one. In this case the number of component per glyph usually is 2, and occasionally 3. Examples of one-operator expressions can be found in 〔 Table 7 〕. After all one-operator expressions of a character set have been collected, a kernel set which has all necessary elements for one-operator expression for any one character/glyph in the original character set can be found. A pictorial presentation of the kernel set of our glyph data base is shown in 〔 Figure 5 〕. This kernel set includes 457 roots,629components,and 1044 kernel glyphs. There are 2173 elements in total. Also, there are 1313 characters in it and the total frequency of usage is 62.60%.

Figure5 ： The Development of Kernel Set (1995~1996)

1996.08.23



Kernel Set ： 2173 elements
Total frequency count ：
6,671,196
System Entropy ： 8.6782
includes ：

- ◆457 roots
- ◆629 components
- ◆1044 kernel glyphs

⊙ 1313 characters
(62.60% usage)

(6)A Descriptive Method of Coding

Kernel set is a closed set. So, it can be coded traditionally by assign a numerical code word to each element. Let us refer this method as numerical coding. Since the total elements of kernel is relatively small, only 2173, it can be easily coded into a 2-byte code space. For the rest 7213 non-kernel characters/glyphs, glyph expressions can be used as their code words. This method is called descriptive coding. Descriptive code word does not require numerical coding space. Besides, descriptive coding is a production system which is capable of taking care additional characters/glyphs to the existing character set.

(7)Optimization

The kernel set in 〔Figure 5〕 is not very efficient, because the total frequency of the kernel is only 62.60% as we mentioned earlier. This percentage can be optimized by including frequently used characters into the kernel set. A frequency distribution chart of our glyph data base is shown in 〔Figure 6〕. In 〔Figure 6〕, it is obvious that including the 1071 characters of the most frequently used category into the kernel will raise the total frequency of usage of the kernel to 97.31%. The price paid is the increase of the numerical coding space to 3243. Which is not bad and can be afforded by any 2-byte coding scheme.

Following the same thought, there are 5 levels of possible optimization of the kernel as listed in 〔Table 6〕. In 〔Table 6〕, the fourth choice is recommended which includes all glyphs of the most frequently used and the frequently used categories. It has 4988 elements and the averaged coding length is only 1.008 times of a 2-byte code.

(8)Re-engineering of Existing Codes

Descriptive coding is compatible with existing codes. When 629 components and 188 non-character roots of the kernel we developed are included into an existing code, the code will have descriptive capability. It can be further optimized by excluding some rarely used characters to save coding space, if necessary. If a kernel set for each country can be derived, then , a CJK unified kernel can be formed , and hence, a CJK unified descriptive code can be constructed.

(9)An Example

For illustration of the descriptive coding method, an example is given in 〔Table 7〕. In 〔Table 7〕, 50 characters/glyphs of the 禸 family from glyph data base are listed. 40 of them are non-kernel characters. Therefore , only 10 kernel glyphs need numerical coding space.

Figure6 ： Kernel Set and Frequency Distribution Chart

1996.10.01

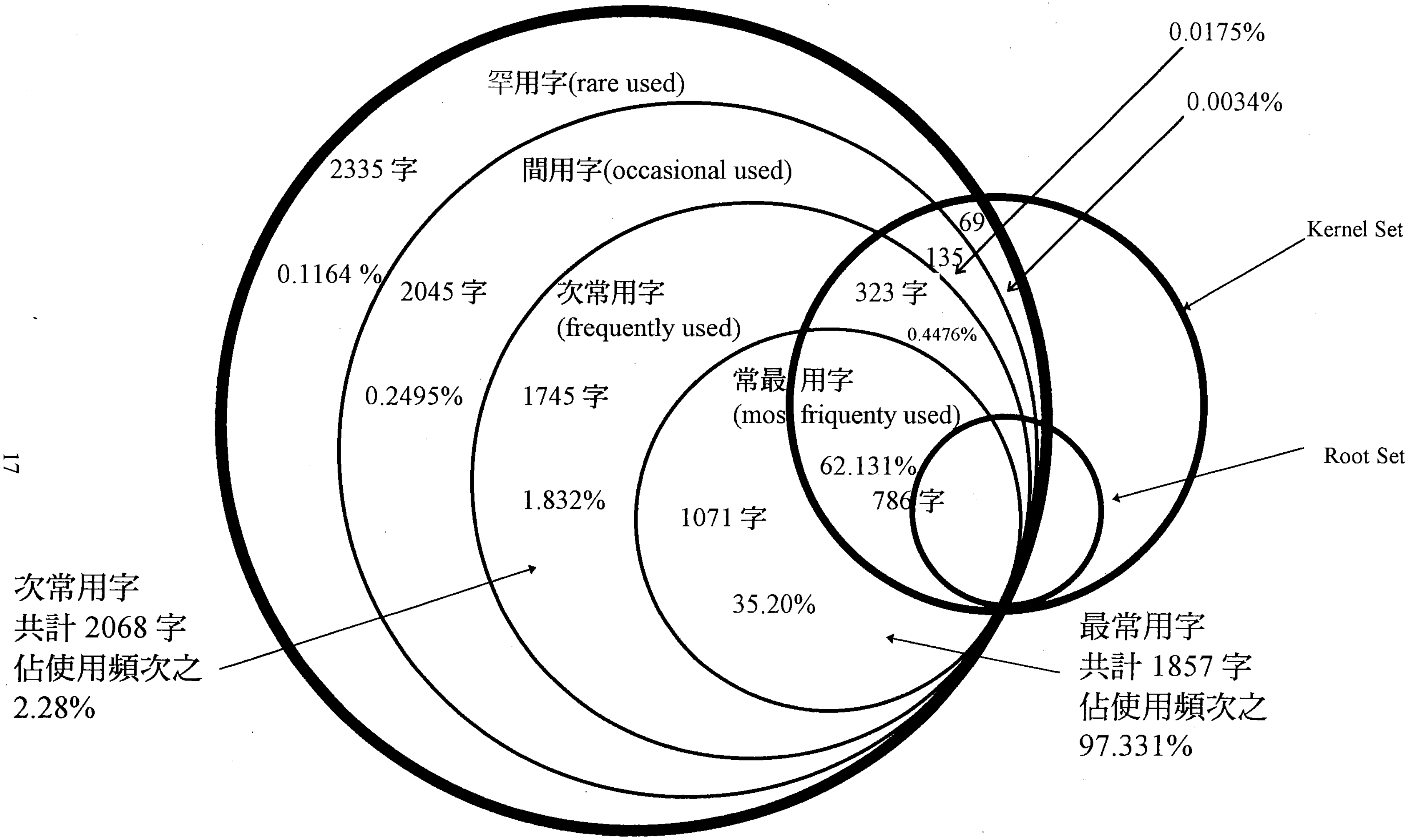


Table 6：《字形系統》各階層之平均字碼長度表

85年8月24日製表

《字形系統》	碼位數	加權平字碼長度	Entropy
1. 字根集	457	1.9+1=2.9	7.3038
2. 種字集	2172		
		(37.43x3+62.57)%=1.7486	8.6782
3. 種字集加常用字	3243		
		(2.2476x3+97.7724)%=1.0452	
4. 種字集加次常	4988		
		(0.3956x3+99.604)%=1.008	
5. 全字集	9346	1	9.1982

註：乘以3是以構字式計算長度
乘以2是以部件序計算長度
在字根性質確定時，如包涵根及++，≀等，則省去聯連符號並不彰響表達之忠實度。

For comparison, the same family derived by Professor 周何 is listed in 〔Table 8〕. These more than 80 characters can all be described by the kernel of our glyph data base. 〔Table 7〕 is a result of graphics decomposition, while Table is derived form traditional Wen-zi-xue. It is clear that they may be merged into one structure by adding proper data structures to the present glyph data base.

(10)Replacement Code

The user defined coding space of numerical coding is no longer needed while descriptive extension is added to the existing code. Therefore, it can be used for other purpose. One possible way is assign this space for replacement codes. The replacement codes are used for replacing some rare used glyphs that happen to be frequently appeared in some specific text. To do so, some markups are needed to specify the replacement. For example, when code xxxx is used to replace a component expression of descriptive coding, the replace relation needs to be specified in the text file. So, a DTD and associated tags of Hanzi file can thus be designed accordingly to facilitate efficiency and information sharing .This is an undergoing topic of our project.

4.Conclusion

A descriptive re-engineering method for existing Hanzi codes is presented in this paper. It is compatible with existing codes. So, this method can be considered as an extension of the existing coding structure, from a closed coding space to an open space. The method presented do follow the guidelines listed in page 2 and hence it provide a way to solve the missing character problem and other drawbacks of the existing codes. Our project is still on going. The sets of components, roots, and kernel glyphs are still under adjustment from tradition Wen-zi-xue and some optimization iterations. But, so far, we think, our work has proved the feasibility of the descriptive coding method to over come the drawbacks of the existing codes.

Table 7：字根「禸」的家族樹

85年8月24日製表

家族樹					字頻次	字根頻次	字根次	構字式		
1・禸					0	4883	50			
	1・⿴禸又				0	920	2	禸◬又		
		1・𤔔			0	920	2	爫⩓マ⩓⿴禸又		
			1・亂		633	633	1	𤔔△△乚		
			2・辭		287	287	1	𤔔△△辛	辛=立⩓十	
	2・禼				0	11	1	卜⩓内⩓禸	内=冂◬人	
		1・⿰釆禼			0	11	1	釆△△禼	釆=丿⩓米	
			1・竊		11	11	1	穴⩓⿰釆禼	穴=宀⩓八	
	3・离				1	1129	17	㐫⩓禸	㐫=亠⩓凶	凶=凵◬㐅
		1・離			871	882	3	离△△隹	隹=亻△△主	主=亠⩓主
			1・籬		10	10	1	⺮⩓離		
			2・灕		1	1	1	氵△△離		
		2・璃			186	186	1	王△△离		
		3・禽			34	49	4	人⩓离		
			1・擒		10	10	1	扌△△禽		
			2・檎		3	3	1	木△△禽		
			3・噙		2	2	1	口△△禽		
		4・漓			2	2	1	氵△△离		
		5・魑			2	2	1	鬼◬离		
		6・褵			2	2	1	衤△△离		

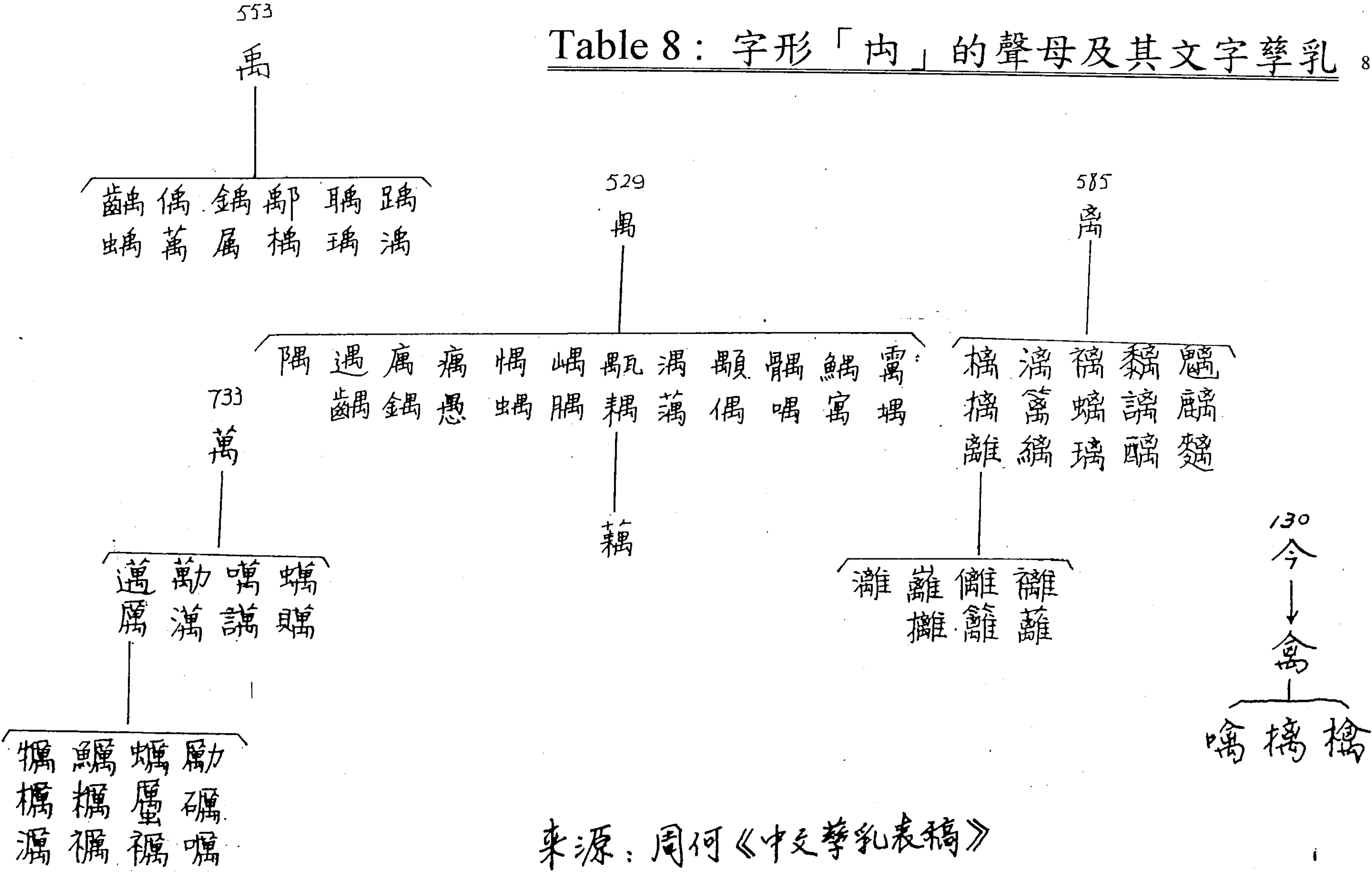
家族樹						字頻次	字根頻次	字根次	構字式			
		7·	摛			1	1	1	扌△△离			
		8·	縭			1	1	1	糸△△离			
		9·	縭$_{2}$(　)			1	1	1	纟△△离			
		10·	螭			1	1	1	虫△△离			
		11·	黐			1	1	1	黍△△离		黍=禾⧊𠆢⧊氺	禾=丿⧊氺
	4·	禹				54	62	8	丿⧊中⧊内			
		1·	屬$_{2}$(　)			1	3	3	尸⟁禹			
			1·	囑$_{2}$		1	1	1	口△△屬$_{2}$			
			2·	矚$_{2}$		1	1	1	目△△屬$_{2}$			
		2·	齲			2	2	1	齒△△禹		齒=止⧊凵从	
		3·	齲$_{2}$(　)			1	1	1	齒$_{2}$△△禹		齒$_{2}$=止⧊凵人	
		4·	踽			1	1	1	足△△禹		足=口⧊止	
		5·	楀			1	1	1	木△△禹			
	5·	禺				1	2761	22	田△△内			
		1·	萬			1576	1826	11	++⧊禺			
			1·	厲		48	197	5	厂⟁萬			
				1·	勵	143	143	1	厲△△力			
				2·	蠣	2	2	1	虫△△厲			
				3·	礪	2	2	1	石△△厲		石=丆⟁口	
				4·	糲	2	2	1	米△△厲			
			2·	邁		29	29	1	辶⟁萬			
			3·	蠆		10	10	1	萬⧊足			
			4·	癘		10	10	1	疒⟁萬			
			5·	勱		2	2	1	萬△△力			

85年8月24日製表

家族樹						字頻次	字根頻次	字根次	構字式			
			6．	蠆		2	2	1	萬⿱虫			
		2．	遇			662	662	1	辶⟁禺			
		3．	愚			131	131	1	禺⿱心			
		4．	偶			108	108	1	亻⿰禺			
		5．	寓			12	12	1	宀⿱禺			
		6．	隅			3	3	1	阝⿰禺			
		7．	耦			3	13	2	耒⿰禺			
			1．	藕		10	10	1	⺿⿱耦			
		8．	喁			2	2	1	口⿰禺			
		9．	嵎			2	2	1	山⿰禺			
		10．	庽			1	1	1	广⟁禺			

Table 8：字形「禸」的聲母及其文字孳乳

85年8月24



来源：周何《中文孳乳表稿》

Table 9：字形資料庫基本字集(1996)

85 年 8 月 22 日製

口	一	丿	亻	十	亠	木	八	土	日	䒑	的	幺	目	丶	氵	月	冂	口	人	又	大	匕	王	厶	宀	言	儿	𠆢	心
𠂉	子	辶	女	攵	扌	小	⺌	艹	冖	是	厂	夕	門	我	寸	不	了	丷	止	隹	几	阝	曰	尸	二	力	乃	夂	乂
有	可	巾	他	罒	龰	工	灬	亼	上	也	⺮	𡿨	在	丰	刂	士	艮	耳	來	廾	𠂆	主	這	國	戈	火	忄	斤	頁
𠂊	車	且	龷	亅	⺍	刀	生	𦥑	ㄨ	中	虫	丟	勹	說	米	⺳	弓	到	個	重	牛	巴	要	以	就	全	犬	你	𠂇
㇇	者	臼	龴	艮	曰	那	耂	勿	⺈	道	业	山	時	丑	冋	裡	馬	丨	和	著	吕	得	家	夕	虍	七	母	三	乍
爲	由	巳	西	豕	里	丆	廿	下	癶	彡	⺊	卩	事	皿	出	礻	看	雨	爿	會	夫	其	弟	水	〢	手	成	永	未
肀	⺄	長	⺫	之	四	午	更	少	皮	兩	面	内	丩	幾	冉	己	先	牜	亥	本	彐	氏	而	民	臣	夬	乙	黑	亡
犭	九	曲	卜	非	衤	乛	厂	身	東	求	酉	卌	已	戊	艮	方	丬	气	丂	舟	疒	弋	凵	匚	丸	田	卬	井	聿
亩	包	及	才	鳥	尹	卝	丯	莫	世	廴	羽	牙	必	丁	吏	太	户	平	少	丗	角	戋	电	内	夊	子	㔾	旡	皿
入	夫	儿	斗	甫	乃	𠃊	申	東	半	缶	𠂢	央	婁	关	非	甘	肀	氺	制	坐	朮	刃	凡	帶	丘	毛	夗	兆	尐
鬼	州	飛	柬	乡	矛	亞	戶	岀	申	片	朩	弗	丙	乚	末	乎	𠂆	垂	夕	毘	川	久	巳	夊	肉	豸	镸	甲	鬥
巨	爾	斥	龍	麗	甲	弌	堇	匚	玉	瓜	耒	产	乘	𦍌	戉	直	尤	巨	⺌	承	爪	屯	兼	𠂤	犮	丈	⺻	瓦	⺗
黽	史	交	乀	束	竹	𠂉	于	卑	卂	巫	肅	屮	丹	隺	互	烏	冊	率	曹	朱	𠄎	典	卵	氶	枼	而	夷	丞	甚
幸	龜	衰	尢	与	巨	市	爽	丐	冘	弔	亙	叉	年	曳	𠘨	鼎	豖	𦣝	傘	由	𠂤	⺇	凸	乖	卅	丰	门	匆	夬
卝	毋	秉	丰	亜	甩	卍	发	𠃊	丏	亜	康	㠯	回	乁	东	具	方	开	东	乜	或	巛	两	丧	弗	甬	奭	氷	氽
冈	乐	世	両	为	孓	业																							

- 共計 457 個字根，依使用頻次排列
- 依校勘後的《中文電腦基本用字》產生

Table 10：字形資料庫基本部件集(1995)

85年8月22日製

糸	彳	冋	广	殳	𦍌	吅	陷	𦥑	古	肓	巨	足	亍	巛	𢆶	冫	开	糸	仆	夋	良	兼	殳	𠫓	且	冊	巠	亽	工
立	疋	击	惠	肰	睘	自	藿	爫	食	易	臼	丵	無	更	班	朋	夅	泉	聿	奐	疋	音	氏	隹	冒	雁	公	匈	豊
留	絡	𢆉	冂	執	殸	口	炊	产	䜌	壴	辰	冂	咼	穷	鼻	畐	虫	产	吉	幽	宝	另	冃	尌	匊	鳥	采	昊	胡
阜	圭	甸	㐬	可	孚	启	删	戴	呈	类	耳	𠬝	舛	厸	卤	共	寅	关	虽	㒼	务	幸	复	召	叨	彑	素	癶	啇
巩	后	家	幾	畿	甬	带	乞	券	段	灵	役	豦	冏	禽	冗	罡	勾	从	罙	圭	赤	俞	丙	爯	育	㔾	尸	皀	夆
枼	风	蒦	盧	蚀	罒	褱	并	咸	氐	攸	遣	昜	圣	彔	余	秝	麻	書	广	产	弓	㐱	扶	仒	冐	並	彭	卧	吳
猒	冒	丞	或	隶	所	皀	殴	珏	盈	缶	尃	䍃	雔	矢	多	臱	冢	縣	甚	丰	巤	㓞	厷	兟	昔	図	光	亏	眘
尿	虐	斥	毄	叀	夗	乇	既	臨	鹵	卬	其	枭	畗	竟	土	时	吉	区	鼠	㗊	業	軋	垔	今	加	劦	㚔	廠	臨
缶	冊	非	卢	壴	衣	爰	幽	癸	盜	次	会	耳	田	書	肙	萬	正	癸	匃	員	参	鬯	忌	局	业	詹	辰	雇	少
兆	虜	隆	臽	素	狠	网	兰	替	売	支	亜	扁	覓	定	凧	曹	亟	步	叔	泰	蚤	瞿	訓	秀	布	呈	維	豈	麦
禺	冠	彐	函	𢦏	季	帛	沙	屰	轉	扁	皿	屏	鶿	尬	胥	虎	乏	塵	墨	旱	勿	秒	亩	肅	錐	鐵	焉	男	聑
朝	則	祟	叒	曼	戳	丵	叢	拜	洰	田	畫	厈	寇	易	及	畕	粦	嵩	奐	肌	言	赢	象	晶	賞	欸	欶	叕	冎
匽	叉	辟	粤	楙	丝	貝	繩	爾	車	谷	汇	西	奮	凡	枚	夒	卜	类	八	占	上	算	宍	处	全	直	興	心	惢
巸	攸	昱	敄	翁	彖	奻	召	凶	侵	帅	公	掌	鴉	樹	卤	垂	由	誉	鵝	区	并	彰	離	東	刺	尋	世	乌	方
耴	品	攸	走	罘	敗	闇	皀	囚	毛	男	匧	翌	厂	販	库	备	孨	杀	么	益	坐	癹	沁	执	獄	疏	墨	昏	每
霝	豩	丌	捶	斷	夹	庸	冏	牀	睿	拑	年	急	畬	彖	岊	砳	妄	伎	鲜	刍	顧	仌	戚	星	条	豸	犇	巨	辮
亭	育	絲	銓	刵	叡	甜	厨	家	畐	伞	觜	去	祭	寻	布	姿	反	韋	齿	麗	毛	瓜	吧	萠	膏	亞	荒	备	摺
盆	卷	骨	垂	脂	會	发	段	匪	犬	雅	肩	此	翁	棠	奎	斤	孚	丞	迷	辈	敦	袁	雁	介	廣	偽	暴	盍	儿
栾	房	会	帚	匈	军	𠫓	卯	亥	戒	尾	留	武	由	刑	旨	索	壹	亢	辥	召	犀	訽	斂	昆	亶	宦	共	交	言
鍊	騙	猷	危	死	奇	鱼	朗	熏	仄	敦	牲	騎	鎚	林	昊	巜	伞	岁	支	斝	傑	僂	僑	洗	紊	彭	介	公	尢
匡	隽	保	乐	嬰	芊	養	經	外	收	盍	珎	道	檗	万	路	醫	亮	灰	启	鳴	备	鼓	亭	麻	舌	总	閣	晉	

- 共計 629 個部件，依使用頻次排列
- 依校勘後的《中文電腦基本用字》產生